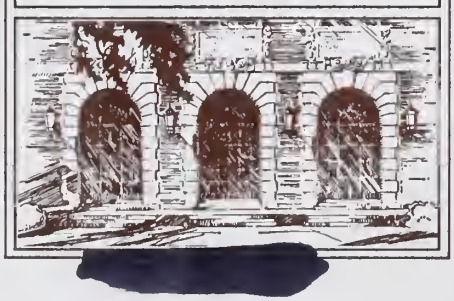
LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
Il6r
no. 746-751
cop. 2



The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

JUL 13 1978

JUL 12 RECD

L161—O-1096

Digitized by the Internet Archive
in 2013

http://archive.org/details/medicaldatamanag751rush

I26n
no. 751 UIUCDCS-R-75-751
Cop. 2
Math

MEDICAL DATA MANAGEMENT

by

ROBERT ALAN RUSH

August 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

LIBRARY OF THE
OCT 6 1975
UNIVERSITY OF
ILLINOIS AT URBANA-CHAMPAIGN

UIUCDCS-R-75-751

MEDICAL DATA MANAGEMENT

by

Robert Alan Rush

August, 1975

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science,
August, 1975

ACKNOWLEDGMENTS

The author would like to express his thanks to Dr. T. T. Chen for his assistance in preparing this document, and to Mr. D. H. Mueller for his helpful advice during the implementations of the medical information system.

TABLE OF CONTENTS

		Page
I.	MEDICAL RECORD MANAGEMENT	1
	The Problem-Oriented Medical Record	5
	Medical Database Organization	12
	MIS Database Organization on Plato	15
II.	DATA DESCRIPTION FOR A MEDICAL DATABASE.	21
	The DDL Specification.	22
III.	DATA MANIPULATION	39
	Implementation Considerations	49
IV.	SUMMARY	51
LIST OF REFERENCES.		56
APPENDIX		57

LIST OF FIGURES

Figure		Page
1.	The Flor of Control on the MIS Project	6
2.	The Patient Data Gathering Process	8
3.	Record Hierarchy	13
4.	Master Directory Organization	17
5.	Possible Access Levels to the Medical Record . . .	18
6.	Master Directory Entry Construction	19
7.	Patient Directory Organization.	20
8.	A Graphic Illustration of the Descendant Relationships A.B and A.C.B.	23
9.	A Graph of A: = (B, (D,E))	24
10.	Structural and Graphic Representations of X: = $\{(\{Y_1, Y_2,...\},Z)_1,(\{Y_1, Y_2,...\},X)_2,...\}$	26
11.	A Graphic Representation of the DESC Function. . .	27
12.	A Graph of DESC(TREE, (MOTHER, FATHER, CHILDREN)) .	29
13.	Structural and Graphic Representations of DESC(TREE,(MOTHER, FATHER, CHILDREN)) DESC(CHILDREN, (CHILD, SPOUSE, CHILDREN))	30
14.	A Graphic Representation of the DATABASE Statement .	31
15.	A Partial Graph of an Example Medical Record Definition	34
16.	A Partial Graph of an Example Problem List Definition	34
17.	A Graph of an Example Progress Note Definition . .	35
18.	A Graph of an Example Medical Record Definition . .	37
19.	A Graph of an Example Database Construction . . .	42

I. MEDICAL RECORD MANAGEMENT

Health-care delivery now constitutes one of this nation's largest industries. Moreover, we have seen a recent acceleration in the demand for health services. The public has come to demand comprehensive medical care as a right, rather than a privilege. However, the costs of health services are increasing at a frightening pace not only in response to the need for advanced facilities, but also to maintain the quality of services. Data collection and maintenance for patient care reviewing, statistical research, and medical communications is a major health-care concern. Health professionals and leaders in medical computing see computer-based information management techniques as the solution to these problems. Dr. J. H. U. Brown declared in November of 1972 that "the U. S. is going to a total health-care system for all 220 million, and it will be based on technology--not on producing that many more physicians."[1] The medical profession is now beginning to open its mind to the support which technology can lend. We are experiencing a revolution in health-care delivery:

> . . .Community health data banks as in North Carolina and Vermont; multiphasic screening in Washington, D. C., Utah, and Ohio; automatic computer analysis of electrocardiograms and electroencephalographs in Pennsylvania, Florida, and Texas. . .,[1]

all reflecting the trend. Computer-based health information systems belong to the emerging technology. With the help of these systems, some of the congestion in medical communications can be relieved. Computers can assist the physician in his clerical functions. Chances for record

error are greatly reduced; standards of input data may be maintained. In all, personnel workload can be lessened, attacking mounting costs.

The medical information system (MIS), in general, can serve two separate functions. The first of these is administrative, or management, information keeping. Computerization is eagerly accepted by physicians and hospital administrators in this role because of its cost-effectiveness. The problems associated with this type of data management are not unique to the health-care profession and, therefore, attract little attention in a discussion of medical information systems. The second function of the MIS is more unique and controversial. It is the collecting and processing of patient care data. The organization of the medical record and database, a problem associated with patient data processing, will be discussed subsequently.

The complete medical information system must be a real-time complex of remote time-shared terminals. These terminals bring the system and, in effect, the information processing powers of a computer to the user. What follows is the establishment of a large on-line accessible file of patient data. In addition, the system should allow communications between all users of the facility.[2]

With the use of an MIS, filing and reporting can be accomplished faster and more accurately. Data collection can be more complete. Interaction with the user can be styled to suit patient characteristics. Computer software allows easy modification of procedure logic, for flexible program development. In addition, data display formats and mediums may be varied according to user needs.

Certain inherent problems exist with a computerized medical information system. It must be extremely reliable. Data stored should remain recoverable despite systems and/or program failure. Also, inspection of any confidential information by unauthorized persons must be restricted.

The Medical Information System being developed at the Regional Health Resource Center (RHRC) and the Medical School (University of Illinois) is presently implemented on the PLATO IV System. PLATO is the object of current research in computer-assisted instruction (CAI) being done at the University of Illinois, Urbana. In its present state, it is a large time-sharing system supporting several hundred terminals. The novel terminal design features a high resolution plasma display panel, touch response, and various other accessories. In addition, the system has proven to be economical and reliable (despite continuing modifications). In all, the PLATO IV terminal and system provide effective means for developing an interactive medical information system. A particular innovative feature of the MIS project is its utilization of the touch response input device in attempting to deal with the problem of the man-machine interface. Another important characteristic PLATO lends to the MIS is its graphic display capability. The displays allow the reduction of large amounts of statistical data to meaningful pictorial forms.

The objective of the MIS has been "to develop systems for large scale acquisition, management, and use of a comprehensive health database

on a regional population, capable of replication and linkage to serve extended population."[3] In pursuit of this goal, the project staff has been engaged in operational testing of the system in development. The feedback from testing sites has been an invaluable input, contributing to continuous project improvement. Indeed, it is imperative that the physician participates in the design of the system, since his acceptance of it will be necessary.

In order to use the MIS, the user must first pass terminal sign-on requirements. Typically, this process is performed once for extended periods of usage on the MIS. Once on the system, a second sequence for user identification is presented to check entry into the medical information system. A selection must be made from a set of existing user groups and also users contained within that group. A password is then required to authenticate the selections.

At this point the user chooses either to access an existing patient record or to create a record. The following phase is one of interaction with the patient directory. There resides identifying data such as name, social security number, sex, etc. and nothing of medical nature. In addition, it is possible for certain persons (those belonging to groups with "owner" access) to view and modify the access rights other groups have to the record. This phase ends when the user chooses to be routed to a routine dealing with some section of the medical portion of the patient record which has been accessed. Existing sections are: Summary, Physical Examination, History, Problem List, Medications, Progress Notes, and Laboratory Results. All of these programs require the

services of the Record Manager, the system's data management facility, for the acquisition of the appropriate medical data associated with a particular patient record. An exit from this stage of the MIS will withdraw the user from the patient record, and he will again be in a position to select a record. Another exit takes him completely out of MIS (see Figure 1).

The Problem-Oriented Medical Record

In the traditional medical record, all observations and information which serve to log the patient's progress are collected according to their respective sources, whether physicians, nurses, or laboratory technicians. What results is an unstructured document--no organization that helps to lead its readers to understanding of its content. In time, a person may cause several of these records to be created by the nature of his illness or perhaps by using more than one institution. This distribution of data can cause the physician to be unaware of relevant facts, or possibly the lack of organization will distort his perception of the patient's medical record. Certainly, at times, a view of the patient's condition is clouded in a deluge of detail. The physician's concerns are not clearly conveyed. What is his thinking behind prescribed treatment and is it related to fact? Many times the tone of the record may indicate difficulties, but one cannot discern which data is relevant to problems which can only be inferred.[4]

In the problem-oriented record, all observations and data are directly associated with a specific and precisely defined problem. Each

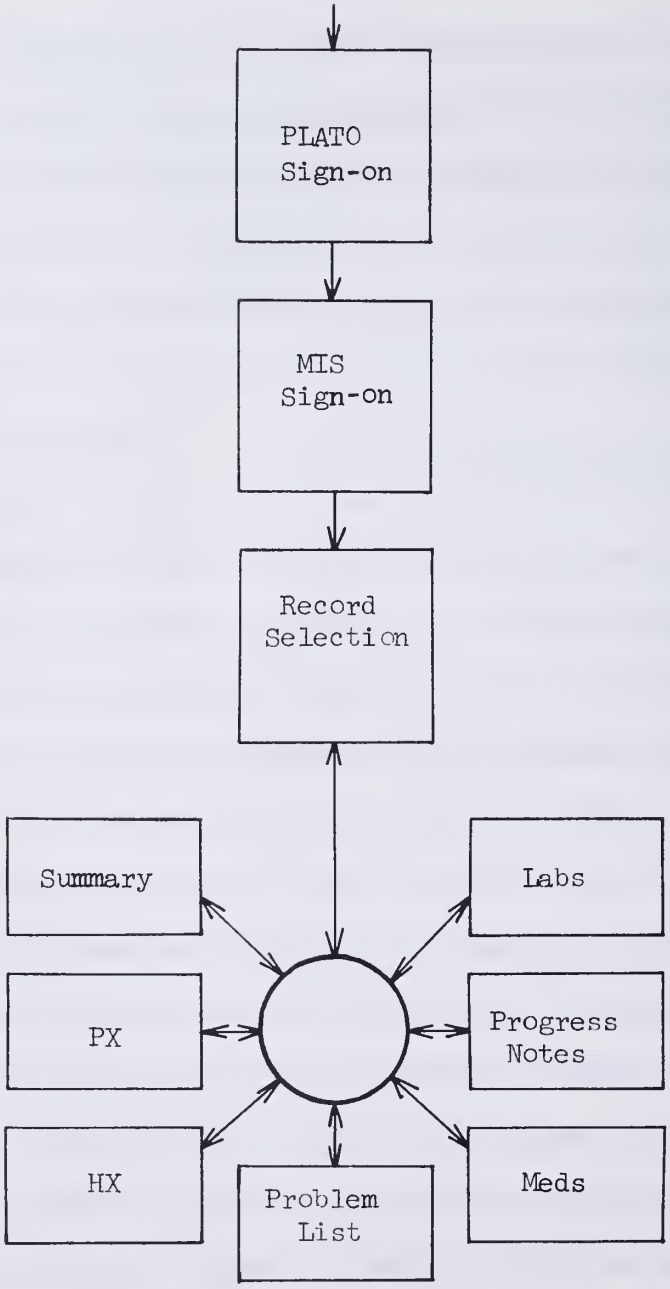


Figure 1. The Flow of Control on the MIS Project.

problem is dealt with in the context of all others. Problem-orientation can be an organizational tool towards clarification and improved communications.

The process by which the single problem-oriented record evolves is where the record's organizational structure is derived. The initial step, upon the patient's first encounter with the physician, is the establishment of the beginning database. With some knowledge to base his judgments on, the physician now determines what specific problems with which his patient is afflicted. And, again referring to available data, a course of action is selected for each apparent problem. In subsequent interviews and tests, more data can be obtained, possibly affecting the list of problems and treatment. Figure 2 illustrates the patient data gathering process.

All phases of the record generation are of course essential, but without the proper collection of data the physician's job may be guesswork or, even worse, decisions made on incomplete or erroneous information. In practice the patient history and physical examination are normally adequate; however, laboratory tests are somewhat arbitrarily performed, at times leaving subclinical disease undetected. There appear to be six types of data in the initial database:[5]

1. The patient's illnesses,
2. His profile and relevant social data,
3. Any present illnesses,
4. The patient history,

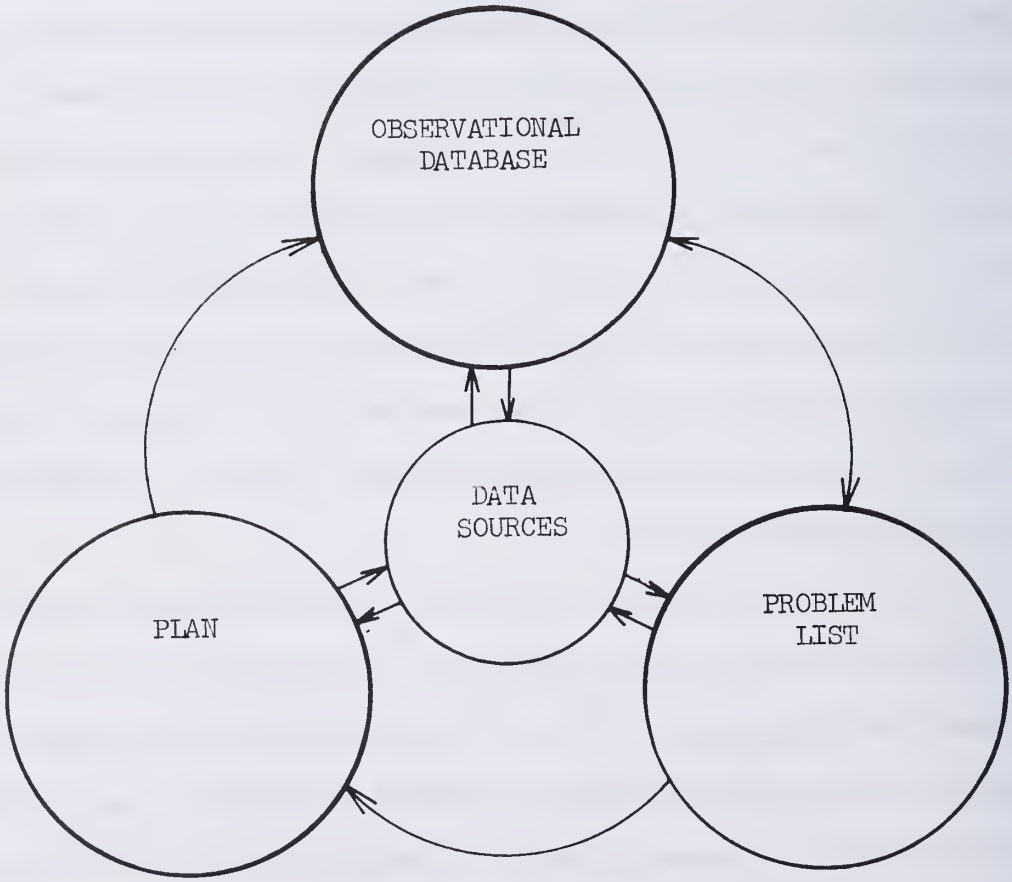


Figure 2. The Patient Data Gathering Process.

5. The physical examination, and

6. Base-line laboratory testing.

Certainly the physician cannot be complete: the history and physical would take forever, and the lab costs would be out of reach. However, some effort should be made for thoroughness. For instance, laboratory and x-ray procedures should be defined and consistently used.

The patient profile must contain a sufficient amount of social inquiries. The physician needs to acquaint himself with the patient's daily routine in order to prescribe treatment realistically. All information about present illnesses should be recorded from all available sources including old records, the patient, and friends and relatives. When a chronic disease is present, symptoms and previous treatment are important. Selection is necessary to reduce data to manageable quantities and is done, hopefully, intelligently and with discrimination.

Already implemented in the MIS on PLATO is a complete branch program medical history. This module is intended to interact with the patient using the touch panel as its primary means of input. The history is a large set of multiple choice questions organized into medically similar sections. Each patient sees a subset of these, chosen according to his own personal data and the responses he makes.

Another phase of data acquisition on the MIS is the physical examination. The physician is able to select input from a large displayed vocabulary or, alternatively, to enter text from the keyboard in random sequence. In addition to the history and physical, is the capability of entering laboratory results.

After the fact gathering process, the physician produces the problem list. It is the record's table of contents and index; its construct determines the quality of documentation. The list must be complete and each problem carefully considered, described, and pursued. The precision and integrity with which problems are formulated will be reflected in subsequent record entries. All problems, past and present, should be noted, whether medical, psychiatric, or social. The physician must avoid making diagnostic guesses and state each element of the list at the level which he understands it. A medical problem may be a diagnosis, physiological, a symptom or physical observation, or perhaps an abnormal laboratory result. If the list entry is social, it should be no less carefully considered and precise. Personality and behavior problems can easily be omitted because they may seem alien to the physician or beyond his scope of knowledge; however, such observations need not be described technically. The list should be expected to evolve as the physician's understanding grows. It may be necessary to alter the phrase which describes the problem. And, what was thought to be a single problem may, in truth, be two (or more). Conversely, two (or more) entries might merge, in time, to be one.[5]

For each problem which the physician determines, a plan is devised. The initial plan is particularly important; it establishes the nature of the data to be obtained and the treatment to be given. Each plan is associated with a problem; omissions may occur when follow-up procedures and medications are mixed randomly. This record organization

allows the reader to see that the physician has reasonably and thoroughly provided for the patient's needs. With time, more detailed plans will be generated.

Before preparing any problem's plan, the patient profile and problem list are first examined. An awareness of the patient's routine and the context of a single problem in the problem list is essential for the prescription of treatment. Each of the variables which enter into decisions concerning treatment should be noted with the plan. A plan may be the collection of data toward a diagnosis or for monitoring a patient's condition. Any questions or doubts, previously inspired, should be investigated with this kind of data gathering. A plan might also be a type of treatment, either therapy or drugs. Any description of medication must be specific. Another kind of plan, one quite often neglected, is patient education.[5,6]

It is the continuous collection of patient data, through the patient's complaints, appearance, lab results, etc., the evolution of the problem list as the database changes, and the corresponding generation of plans which constitutes the iterative production of progress notes. This process is organized in the S-O-A-P (Subjective-Objective-Assessment-Plan) format of the progress note. S refers to data collected from patient complaints and profiles and O to laboratory and physical examination information. A is that explanation of contributing factors mentioned before and at times a diagnosis. Each problem may have several associated progress notes, and every note is linked to a specific problem.[6]

All elements of the problem-oriented medical record have now been discussed. The pieces fit a hierarchical picture; the problem list forms the record's backbone. Each problem gives way to a set of progress notes, each of which has the SOAP structure. The only addition to this is that some of the elements of the initial database, such as the history, reside alone being of importance to all problems. This definitive formatting allows the record to be computerized and yet does not require it. Figure 3 pictorially represents record hierarchy.

The MIS user, after selecting a patient's record, may choose to deal with any portion of the medical record, and subsequently, branch between the routines related to the parts.

Medical Database Organization

The organization of a medical database involves imposing a structure over all elements of health data, defining how these elements are related and typed, and providing capabilities for securing confidential data. The traditional view of this database has been that of a set of patient records, sometimes grouped according to family. In order to more generally describe the database, it might be advantageous to present a different view, that of a collection of data elements and data structures. The record, as we think of it, is one special kind of these structures. Furthermore, each element or structure may be given a type which indicates the nature of the data it represents. Therefore, history data may be distinguished from laboratory results. A single

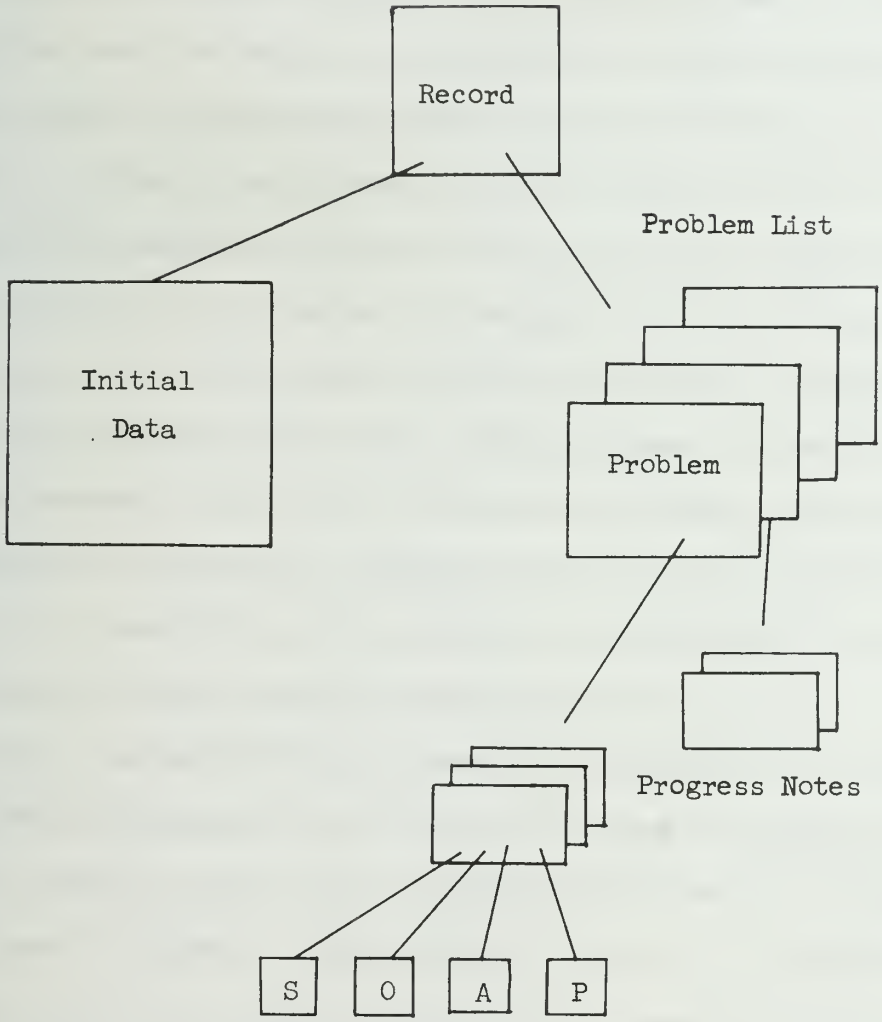


Figure 3. Record Hierarchy.

problem would represent a structure over its description and progress notes.

The medical database has a variety of users requiring a range of accessibility. The physician is naturally interested in the records of particular patients, possibly some specific portion of a record. At times, however, he may wish to access information elsewhere in the database in the process of reviewing the record. The researcher might investigate the histories of all patients, and not need or have authorization to view other confiential record parts. Two directions of access, vertical and horizontal, exist and the organization of the database should support both accesses. The vertical movement takes the user from the structure to its components. Horizontally, he moves along the lines of "like" data, or from structure to structure.

With this general hierarchical organization, the current medical record could reside in a super-structure of somehow related records and be composed of a sub-structure depicting the components and relationships of the problem-oriented medical record. Data elements, as well as structures, must be constructed flexibly. In practical use, two separate history elements may be quite unsimilar data although each relates to a patient history.

Security presents a formidable problem in the design of the medical database. Certainly protection begins at the terminal, and methods incorporated here may affect internal checks being done by the database system. Physical access to the terminal can be restricted,

along with the system log-on. The log-on may be accomplished by hardware device techniques (magnetic cards, fingerprint recognition, etc.) or by password authentication of the user identification. In either case, the database manager must be acquainted with the user by his name, or some equivalent descriptor, for further security checking.[7]

The capability should exist for locking out unauthorized users on each type of data. A lock on a structure would imply some particular level of access for all components of the structure.

Protection should be based on the granting of access rather than excluding it. This conservative approach is more fail-safe, and design errors result in an implicit lack of permission. Additionally, the design of security procedures should not depend on the ignorance of potentially malicious users. Methods must withstand the knowledge of design techniques, and data remain secure.[8]

The local MIS utilizes a vertical database, a file of patient records for access by record. The following description of that system will reveal that it does not fulfill the previously stated requirements for medical database organization.

MIS Database Organization on Plato

On the PLATO system, disk memory space is partitioned into "lessons" and "datasets." All space is allocated by "parts," a specific physical size, namely seven blocks of 322 words each. A word on the CDC hardware being used is 60 bits long or, equivalently, 10 characters.

The lesson type data space is, for the most part, written with the source code of PLATO's high level language TUTOR and could, in this sense, be called program space. The capability also exists for defining an in-lesson work area referred to as a "common." No disk input or output is necessary to access the common, as it is loaded into core memory with the execution of a TUTOR command.

The dataset must be accessed via disk I/O. Reads and writes are done in block quantities by TUTOR lessons upon execution of the appropriate commands. Whereas only one common may be associated with a lesson, any number of datasets could be accessed during program execution.

The RHRC MIS on PLATO provides for vertical accessibility to the database; that is, data elements may only be encountered through the access of a single patient record. The medical records are filed according to their respective patient index numbers, assigned at the time of record creation. All patient data is organized under the structures of the master directory and patient directories.

The master directory index list is a set $[A_0, A_1 \ldots, A_n]$ where A_i is an address of a disk block which has been allocated as a master directory block. Each such block contains potential space for four master directory entries (MDE), each associated with a patient record. In particular, A_i is the address of the block where MDEs for records $4i$, $4i + 1$, $4i + 2$, and $4i + 3$ reside. The MDE takes an 80 word space and therefore its offset from the beginning of the block for some record j is given by (j mod 4) x 80 words (see Figure 4).

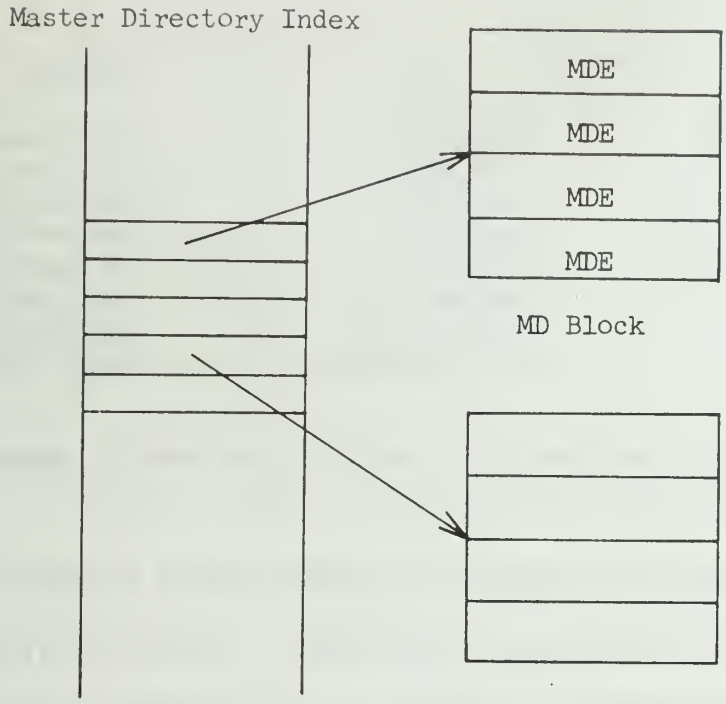


Figure 4. Master Directory Organization.

The identifying data, such as the patient's name, social security number, sex, and next of kin, is kept in these 80 words. Also in the entry is the record's access list. Each element of the list is the access level which has been assigned to the group associated with that position in the list. Each level gives that group two particular types of security status with respect to the identifying and medical data. The special status of "owner" allows the group to modify levels of access for other user groups on that record (see Figure 5).

Another portion of the MDE is set aside for trace information. The user who caused the record to be created and the last user to view or edit medical data, along with the dates and times of each, are stored here.

Access Level	ID	Medical
0	no access	no access
1	read only	no access
2	read only	read only
3	read only	read-write
4	read-write	no access
5	read-write	read only
6	read-write	read-write
7	owner access	owner access

Figure 5. Possible Access Levels to the Medical Record.

Finally, the address of the block which has been allocated as the patient directory belongs in the MDE. The patient directory is an index into the remaining record data. A discussion of it requires a definition of the term "data item" in the context of the MIS on PLATO. A data item is a piece of data which is referenced under the data management facilities by a single name, may have a size of either 5, 10, 20, 40, 80, 160, 320 words, and is allocated for the use of the user interface programs. In order for the history module, for example, to utilize any dataset space in the patient record database, it must request the creation of a data item for a particular record (see Figure 6).

The patient directory consists of three major entities: the block list, the item list, and free space list headers. As data space is required for a patient record, disk blocks are allocated to it and recorded (as addresses) in the block list. According to this scheme, no two records own data in a common disk block. If a 10 word data item

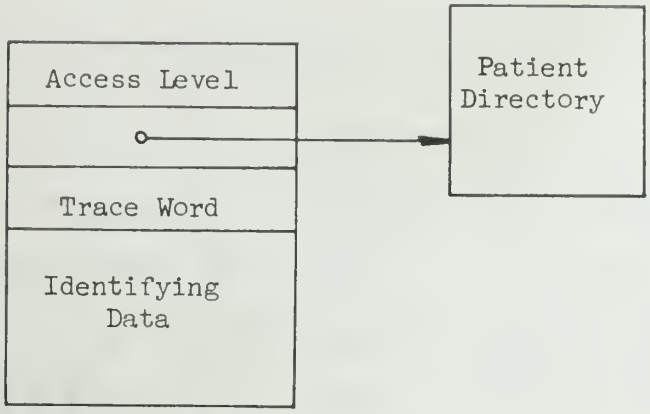


Figure 6. Master Directory Entry Construction.

is requested and no free position of that size (or larger) already exists in allotted blocks, a new block is allocated.

The item list is a sequence of previously created data items. Each element of the list associates a 30 bit name, a size, an entry in the block list, and a relative location within the block with a data item. During execution of the MIS this list figures heavily in data manipulation functions.

Of the blocks which belong to a patient record, the space which is not in use is broken into pieces of 5, 10, 20, 40, 80, or 160. Each piece is threaded onto a linked list of its size whose root is stored in the patient directory. There are, then, six of these roots, or free space list headers. This technique serves to reduce the amount of internal fragmentation of memory, and thus resources are better utilized (see Figure 7).

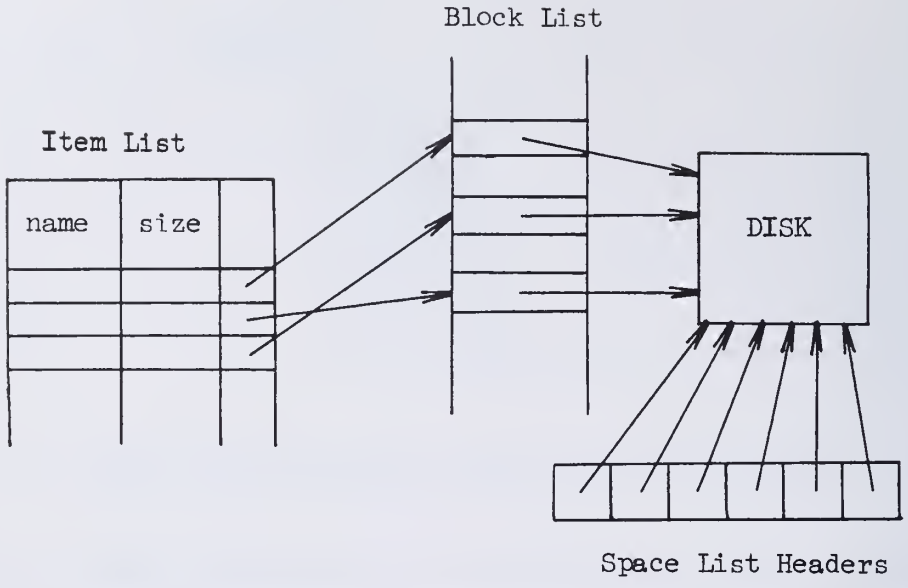


Figure 7. Patient Directory Organization.

II. DATA DESCRIPTION FOR A MEDICAL DATABASE

In order to present a general scheme for the organization of a database, one must be able to reference elements and areas of that database, and to address the structures and special relationships which may exist. Thus, an instrument to facilitate the communication is a necessity. A data description language (DDL) allows the user to construct a multidimensional picture of the database. Any position in the structure may be referenced by the roads which lead to it. The DDL presented here is intended to demonstrate a method which can be used to describe the medical database and meet its particular requirements of accessibility and security.

The term "data item" will be used (more generally now) to denote elementary data as opposed to a data structure. The data item can be represented by some name given in the DDL and indexed as a part of an "item array." The item array is essentially a naming convention, denoting several like items existing at the same position in the data structure. ITEMA (I) will indicate the Ith member of the item array ITEMA. Again, items are elementary and thus have no associated structure.

To accomodate the various users of the database, it would be necessary to define a different view of the data for each mode of access. A structure is created over the collection of data items, choosing those items which are relevant to a particular user and relating them according to the user's need. Several of these structures may be necessary for the physician, lab technician and clinical researchers.

Although only a single real database exists, it appears to be more than one to the users. In essence, there are multiple mappings of the database. Much of the information contained in each map is the relationships between items and the security maintained over them. Security controls may differ over the spectrum of names an item may have.

The database description may be evisioned as a directed graph. Each name in the description is represented by a node of the graph and every relation which is defined corresponds to a path or edge. Those nodes which have no paths leading away are the item arrays. Those that no paths are directed toward reflect possible entry points for access. It is possible then to store the description in a similar fashion for reference by the data management system. If this scenario is turned around, the DDL may be thought of as a means of describing the graph and it remains to be shown how such a structure will be defined. It is important to note that the description is not the database itself and, therefore, iterations of structures which are possible need not exist in the graph, but only the knowledge that they may occur is necessary.

The DDL Specification

Let us define a special relationship "descendant" as follows: a node B is the descendant of a node A if and only if there exists an edge from A to B, or from A to some node C such that B is a descendant of C. If node B is an iterative form (its data type applies to a

vector of data groups) and may be indexed B(1), B(2), etc., and is a descendant of node A, only a single node denotes the form, representing the relationship between A and all members of B. We will also call the node B the "direct" descendant of node A if there is an edge from A to B (see Figure 8).

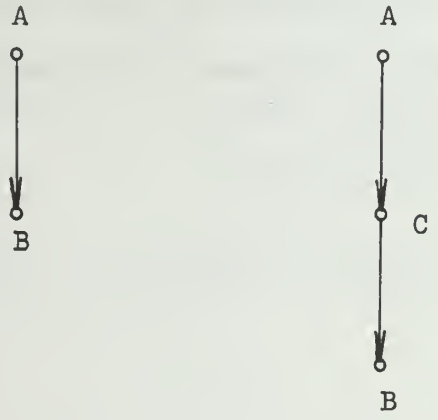


Figure 8. A Graphic Illustration of the Descendant Relationships A.B and A.C.B.

Formally a structure will be defined as a node which has descendants. All of the "real" data, originating from the user, which is contained in a structure resides in its descendants. Recursively this implies that only the terminal nodes, data items, are in fact user-supplied data, the rest serving to relate these items.

As a convenient way to reference nodes of the graph, a node B which is the direct descendant of node A may be called A.B. Likewise,

if C directly descends from B, it is A.B.C, B.C, or A.C, as long as ambiguities are resolved. In order to speak about actual elements of the database which obey these relationships, it might be necessary to cross several levels of indexing. For example, whereas the relation A.B.C has been defined, some element which belongs to the structure could be A(8).B(2).C(3) (which is quite distinct from A(8).B(3).C(3)).

Each node of the description graph will be defined in terms of its direct descendants. The node A whose direct descendants are B and C is formally defined:

A: = (B,C)

where the symbol ": =" means "is defined as." If node C is in turn stated:

C: = (D,E)

then it may be said:

A: = (B, (D,E)).

(See Figure 9).

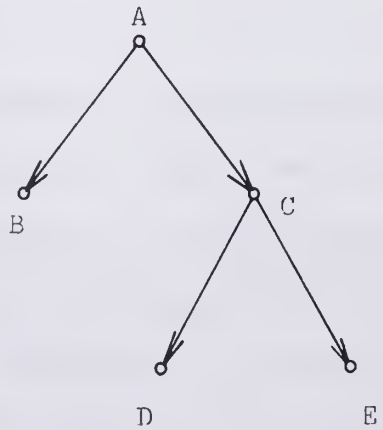


Figure 9. A Graph of A: = (B, (D,E)).

The symbol "Item" will be reserved to denote the fact that the node has no descendants, as in:

B: = (Item).

Therefore,

A: = ((Item), (D,E)).

If the node is representative of an iterative data type and may be indexed, it will be defined as follows:

$X: = \{X_1, X_2, X_3, \ldots\}$.

Each X_i obeys the same relationships that apply to node X; namely, if X has descendants Y and Z, the same is true for each X_i. Therefore, it could be stated:

$X: = \{X_1, X_2, X_3, \ldots\}: = \{(Y,Z)_1, (Y,Z)_2, (Y,Z,)_3, \ldots\}$.

In addition, if

$Y: = \{Y_1, Y_2, Y_3, \ldots\}$,

then,

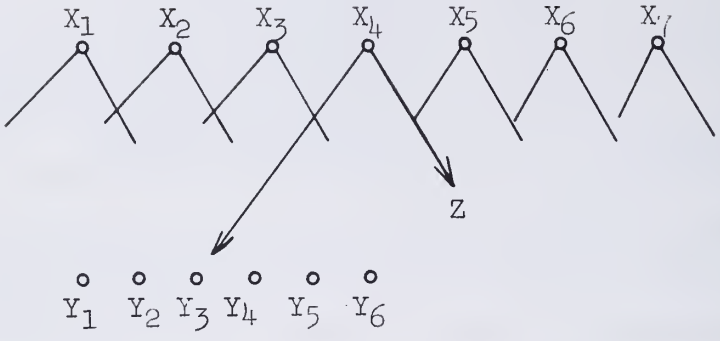
$X: = \{(\{Y_1, Y_2, \ldots\}, Z)_1, (\{Y_1, Y_2, \ldots\}, Z)_2, \ldots\}$.

(See Figure 10). Note in particular that if Y has no descendants, then

$Y: = \{(\text{Item})_1, (\text{Item})_2, \ldots\}$

and is what is referred to as an item array.

A pair of functions will be necessary to facilitate the presentation of the DDL. The first of these, DESC, makes it possible to describe each node. The first operand is a node and the second is an ordered n-tuple to be associated with the first operand as direct descendants (see Figure 11). For example, the operation



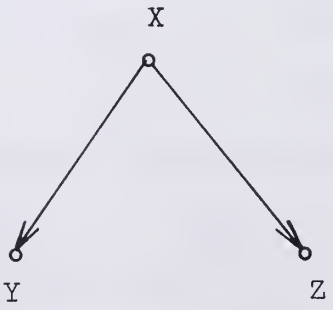
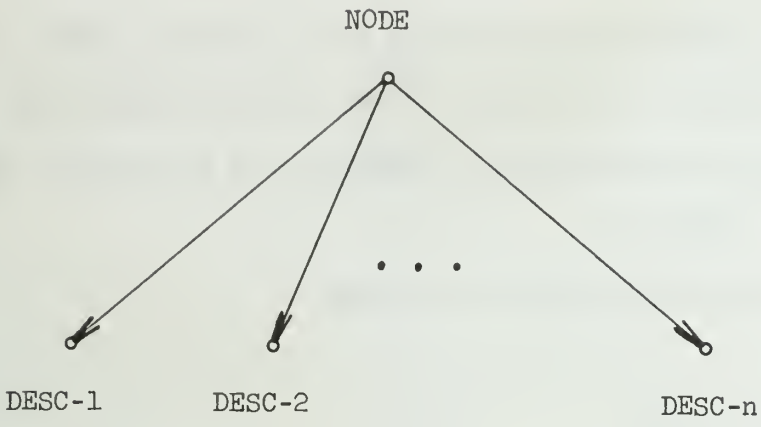


Figure 10. Structural and Graphic Representations of $X: = \{(\{Y_1, Y_2, \ldots\}, Z)_1, (\{Y_1, Y_2, \ldots\}, X)_2, \ldots\}$.



DESC(NODE,(DESC-1,DESC-2,...,DESC-n))

Figure 11. A Graphic Representation of the DESC Function.

DESC (K, (L,M,N))

produces the following relationship:

K: = (L,M,N,).

Similarly

DESC (M, (Item))

results in:

M: = (Item).

The following sequence of operations:

DESC (A, (B,C))

DESC (C, (D,E,))

DESC (B, (Item))

produces:

A: = (B,C)): = (B, (D,E)): = ((Item), (D,E)).

From the above statement about the node A, one can discern that A.B is an item and A.C.E is a relation which exists in this structure.

Another function which will be used is ARRAY. Its single operand is a node. The effect of the function is that its operand is designated as being iterative. Therefore, as an example, the operation

ARRAY (X)

would allow the statement to be made:

$X: = \{X_1, X_2, X_3, \ldots\}$.

The two operations

ARRAY (A)

DESC (A, (Item)

specifies that A is an item array.

With the two functions that have been presented, the graph may be constructed to describe the relationships between parts of the database and data types can be designated as iterative for convenient reference. To illustrate the uses of each function, let us construct a simple family tree structure. The database will encompass all descendants of a particular pair of people. The node which gives rise to the structure, the entry point, will be called TREE and defined thusly:

DESC (TREE, (MOTHER, FATHER, CHILDREN)).

(See Figure 12). The mother and father are not structural, and therefore stated as:

DESC (MOTHER, (Item))

DESC (FATHER, (Item)).

The children, however, may have descendants and develop structures of their own. So they are defined:

ARRAY (CHILDREN)

DESC (CHILDREN, (CHILD, SPOUSE, CHILDREN)).

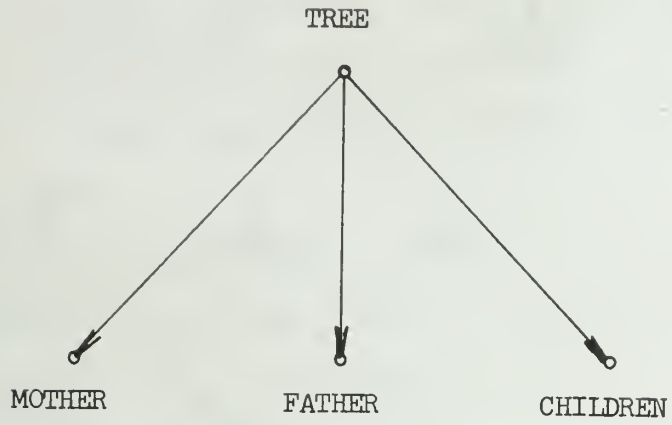


Figure 12. A Graph of DESC(TREE, (MOTHER, FATHER, CHILDREN)).

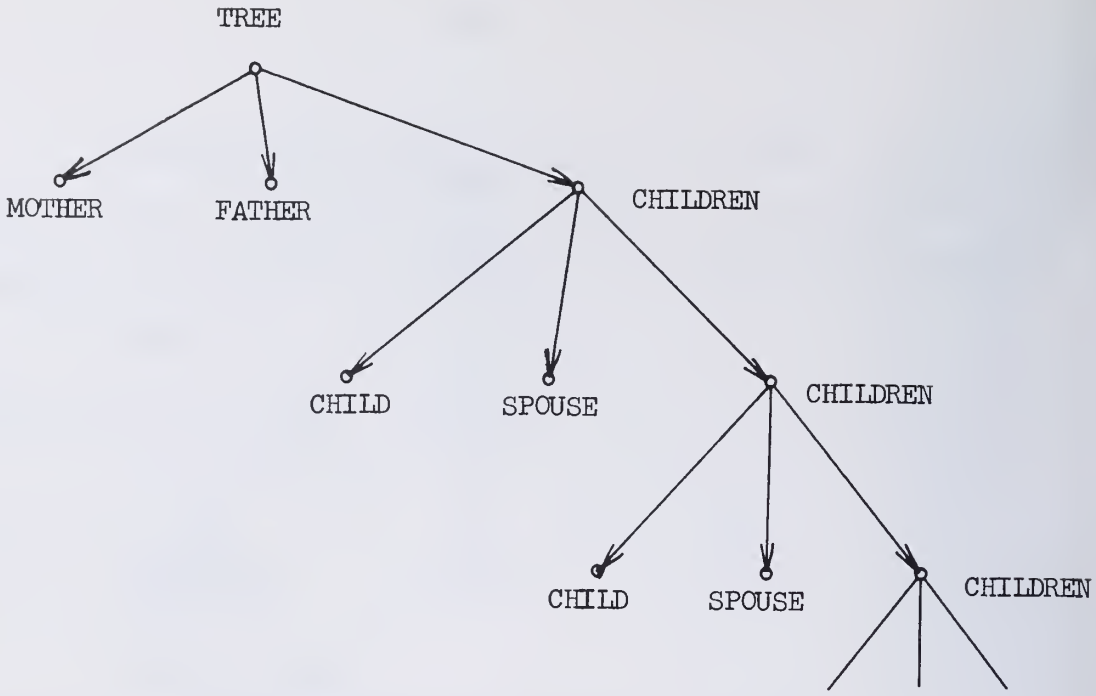
In addition:

DESC (CHILD, (Item))

DESC (SPOUSE, (Item)).

(See Figure 13). Note that the definition of CHILDREN is recursive. By the meaning which has been given to the functions, it would indicate that the relationship TREE.CHILDREN.CHILDREN.SPOUSE is possible. In the database, if the data existed, one would reference the first child's second child's spouse by TREE.CHILDREN(1).CHILDREN(2).SPOUSE. The levels of indexing occur at nodes which are operands of the ARRAY operation.

The statements of the DDL are designed to utilize the given functions in such a way that the database design becomes easier and more convenient and to help in creating the internal garaphic representation. There are two basic statements, DATABASE and SET. The conceptual



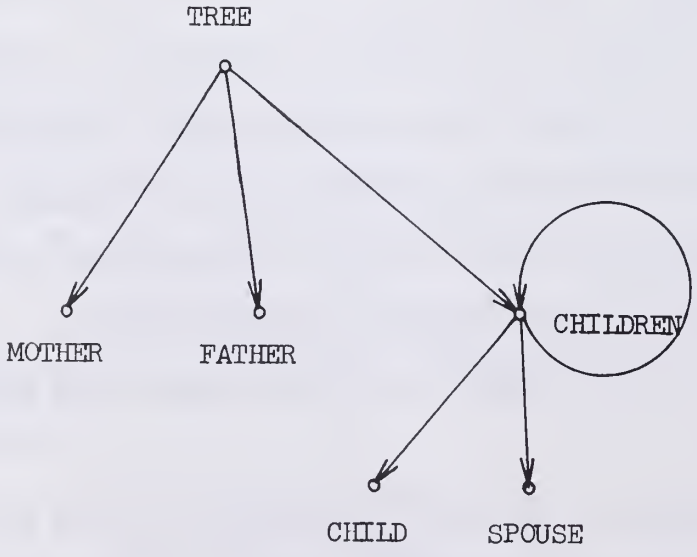


Figure 13. Structural and Graphic Representations of
DESC(TREE,(MOTHER, FATHER, CHILDREN))
DESC(CHILDREN, (CHILD, SPOUSE, CHILDREN))

purpose of the DATABASE command is to create an entry point into a structure designed to meet the needs of a certain set of users. The SET is the structure building statement capable of molding the relationships which are necessary.

The form of the DATABASE statement is as follows:

DATABASE: <entry-name> (<desc-1>, <desc-2>,..., <desc-n>)

Its functional equivalent is given below:

DESC (<entry-name , (<desc-1>, <desc-2>,..., <desc-n>)).

(See Figure 14). In essence, a name is associated with a particular route of access to the database. As was shown previously, a descendant may be referenced through the entry name. Such statements, along with those defining the nature of the descendants, will describe all modes of acquiring data. No descendant may be specified as an entry name also, this violating the intentional use of an entry point.

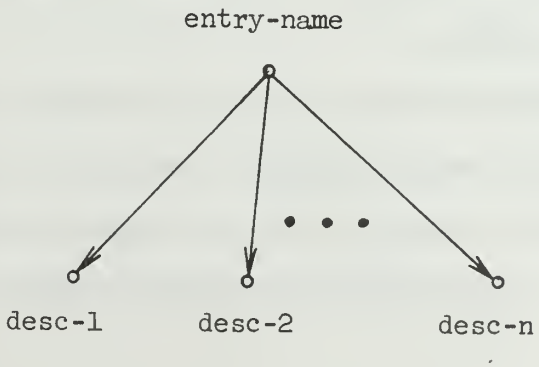


Figure 14. A Graphic Representation of the DATABASE statement.

The SET statement is of the form:

SET: <set-1>, <set-2>,..., <set-p> (<desc-1>,... <desc-q>)

Its functional equivalent is given below:

```
ARRAY (<set-1>)
    .
    .
    .
ARRAY (<set-p>)


DESC (<set-1>, (<desc-1>,..., <desc-q>)).
    .
    .
    .
DESC (<set-p>, (<desc-1>,..., <desc-q>)).
```

Because this statement is the only method of describing data structures other than the DATABASE command, it is evident that all structures below the entry point level are potentially a set of iterations. However, this is not a problem from the user standpoint, since the single data type without multiple copies may be considered a single iteration of the structure. The user will manipulate the data according to his wishes and extra, unnecessary capabilities should not be detrimental to his use of the database.

Notice that the SET provides the ability for defining like structures under different names. Certainly this could be accomplished using several SET statements; however, implications may be derived from this form of the SET according to implementation and will be discussed subsequently.

Data names which have been referenced in DATABASE or SET statements and yet have not been designated as an entry or set name, are implicitly defined according to the functions (where X is such a data name):

ARRAY (X)

DESC (X, (item))

and thus are item arrays. This is also equivalent to

SET: X( ).

Let us construct an example which will show how the statements presented might be used to describe the medical database. First, consider the view of the physician wishing to access individual patient records. The following statements establish the root of this structure and basic formation of the record index:

DATABASE: MEDRECS (RECORDS)

SET: RECORDS (ID, HX, PX, LABS, MEDS, SUMMARY, PROBLIST).

(See Figure 15). The data name ID represents patient identifying data, HX the patient history, PX physical examination data, LABS initial laboratory results, MEDS previous medications, SUMMARY the record summary, and PROBLIST the problem list. Of these, only PROBLIST is a structure and thus the only one to be defined as a set name. The problem list may be described as follows:

SET: PROBLIST (PROBDESC, PROGNOTES).

(See Figure 16).

PROBDESC represents the problem description and PROGNOTES the vector of progress notes associated with each problem. To define the

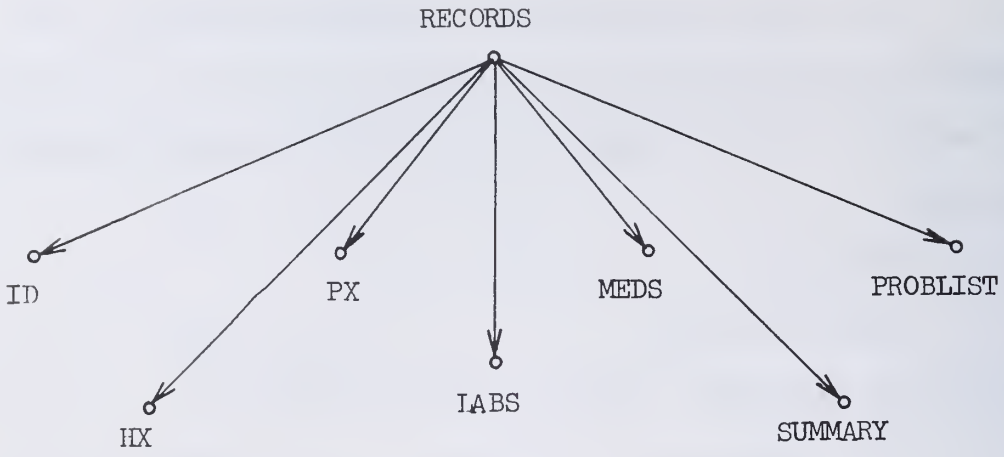


Figure 15. A Partial Graph of an Example Medical Record Definition.

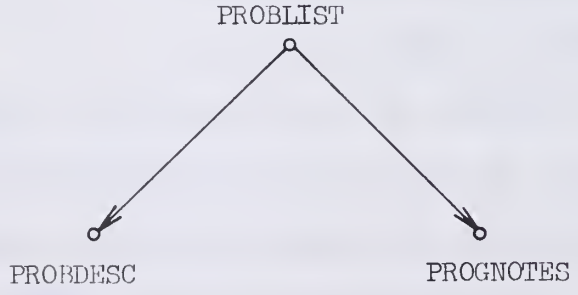


Figure 16. A Partial Graph of an Example Problem List Definition.

latter, another SET statement must be included:

SET: PROGNOTES (S,O,A,P)

(See Figure 17). The descendant list here refers to the S-O-A-P format of the progress note. Under this description of the database, the subjective segment of the third progress note of the second problem in

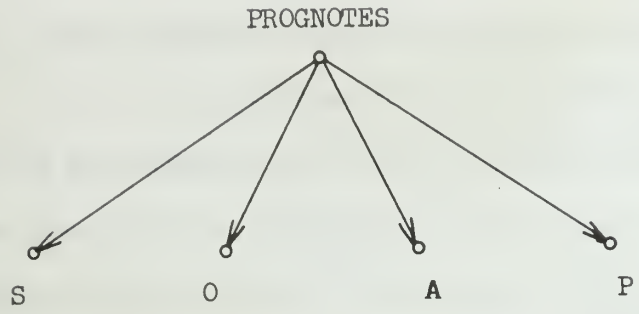


Figure 17. A Graph of an Example Progress Note Definition.

the fourth record is referenced MEDRECS.RECORDS(4).PROBLIST(2).PROGNOTES (3).S(1). Although it may be intended that S be a single data item, it is implicitly an item array. From this point on the first element of array data may be called either by the subscripted name or by the name alone. Therefore, in the above example, both S(1) and S are acceptable.

Let us imagine a user who wishes to investigate the progress notes which exist in the database without a need for knowledge of other data related to progress notes, such as the problem list. The SET statement previously given as:

SET: PROGNOTES (S,O,A,P)

is modified to:

SET: ALLNOTES, PROGNOTES (S,O,A,P).

In addition, the statement:

DATABASE: INSPECT (ALLNOTES)

is included. As data of this form is added to the database, it is created under both the names ALLNOTES and PROGNOTES. The former name allows an easier access to the data, for example INSPECT.ALLNOTES(15), but without the advantages of the structure which pinpoints the data's membership in the organization of the record system. The set of progress

notes may also be organized under different criteria, and reside in structures unrelated to record organization.

Another option on both the DATABASE and SET statements is the "LOCKS" suffix and allows for security controls over any particular type of data. The complete form of each of the statements of the DDL is:

DATABASE: <entry-name> (<desc-1>,..., <desc-n>) [LOCKS]

SET: <set-1>,..., <set-p> (<desc-1>,..., <desc-q>) [LOCKS].

The effect, in either case, is to associate an additional descendant with the node, one which is a data item, specially created for a security lock list. This lis provides certain types of access for various users over the data structure which is defined in the statement specifying LOCKS.

The following set of DDL gives a possible specification for a family patient record system:

DATABASE: MEDRECS (FAMILIES, RECORDS) LOCKS

SET: FAMILIES (FAMRECS, FAMINFO)

SET: RECORDS, FAMRECS (ID, HX, LABS, SUBREC)

SET: ID ( ) LOCKS

SET: HX ( ) LOCKS

SET: LABS ( ) LOCKS

SET: SUBREC (PX, MEDS, SUMMARY, PROBLIST) LOCKS

SET: PROBLIST (PROBDESC, PROGNOTES)

SET: PROBNOTES (S,O,A,P)

(See Figure 18.)

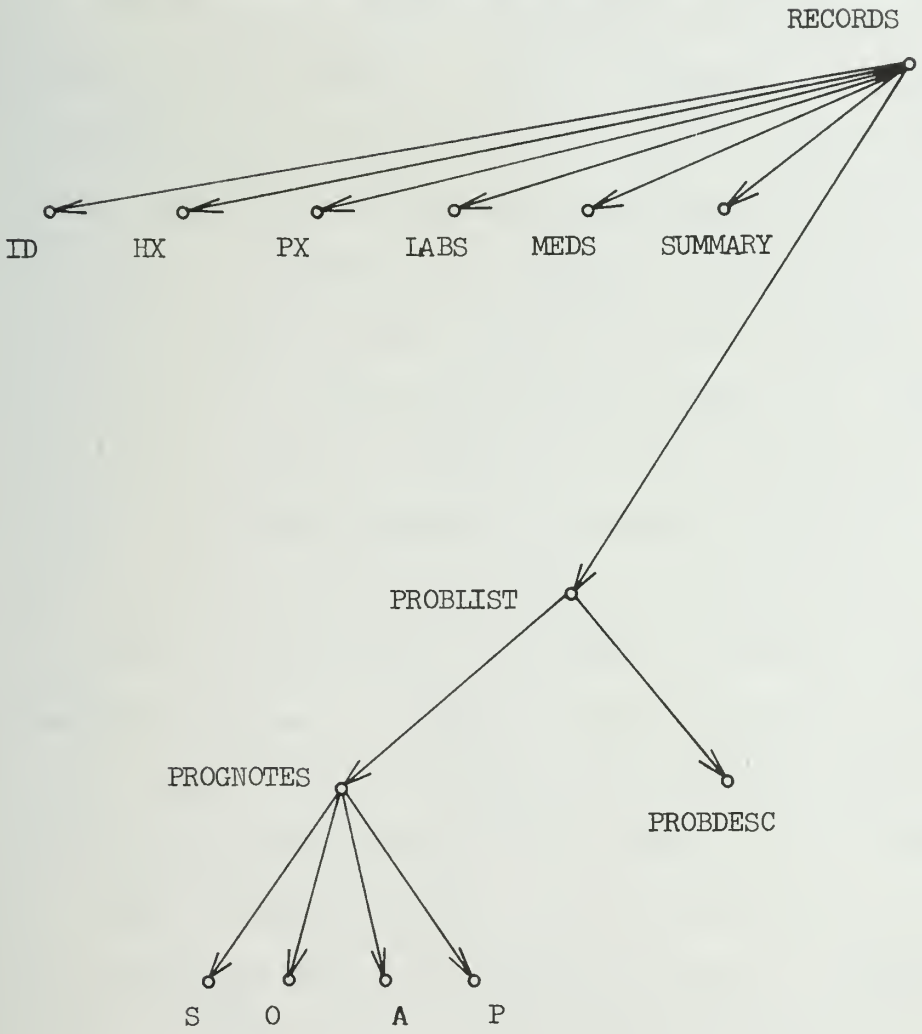


Figure 18. A Graph of an Example Medical Record Definition.

This view of the database allows records to be included in a family, although they may belong to the set RECORDS also. Various parts of the record may be secured separately so that different users of the record may access only the regions which they are authorized for.

III. DATA MANIPULATION

A DDL has been presented for the description of a database, its structure and the relationships between the data which belong to it. It is necessary, with this organizational scheme, to reference all elements of data precisely. The data management system utilizes the advantages of the organization when performing basic manipulative operations. This system must directly interface with the database. The functions which comprise the interaction form the central module of the program set using the data. Among the tasks of the module are the input and output of data to disk storage from/to user areas. In an interactive environment, where each terminal must have attached work regions, input/output functions are designed to pass data between user-specified locations and memory chosen by the data manager. A large set of buffers must be available, and subsets allocated to a user task. Therefore, each user is working with an assigned array of buffers plus his work space. More specifically, the application program deals with the data management system and thus provides the user interface. While this type of program uses only its work space directly, the data manager needs the buffer array for a greater efficiency in processing service requests. However, in order to reference user-accumulated data, all attached memory areas will be envisioned as an array denoted by the symbol "#."

Certain operations may be done specifically at the request of the application program. These are the data manipulation functions

which deal directly with user buffers and database elements and structures. To illustrate what types of operations should be possible for the using program set, an "assignment" type statement will be necessary. The form of the statement will be: target ← value. The target represents some location in the database structure or data buffer area where some form of data is to be placed; the "value" is that form, denoted by a similar notation as the target, except that it must designate existing information as opposed to a hypothetical position in the structure. The symbol "$" will be used as the last created index in an array, as this numeric value must necessarily be available for continued addition to indexed sets. Also, besides being able to discuss specific indices, positions between members of an array are important for insertions and must be referenced. The signs "+" and "-" will make it possible to address these locations; for example, "2+" will denote the slot after member two (and before three), and likewise "1-" refers to a place in the array immediately before the existing first member. It is clear that since this notation does not represent previously created data, such symbols will only be seen on the target side of the assignment statement. The above conventions will make it possible to discuss the basic functions required by the accessing routines for the dynamic structuring and modification of the database.

Let us first consider a hypothetical database description, and show how data may be added to it, and manipulated:

DATABASE: ADATA (AA, AB, AC)

SET: AA, BA, (X1, X2, X3)

SET: AB (Y1, Y2)

SET: BB (Z1, Z2)

DATABASE: BDATA (BA, BB)

SET: X2 (X20)

(See Figure 19).

Note that the ordering of the DDL statements is unimportant.

It seems natural to first discuss the creation of data elements to the database. The application program through its interaction with the user, accumulates some amount of data it wishes to be known as a single item. Several variables, the state of the database, the identity of the user, the stage of interaction which the user is in, all affect the placement of that item and the name or names which will be associated with the data. The following is an example of the operation to be performed:

ADATA.AA (2).X1 ($+) ← #(3)

Specifically, the notation says that the third user work space is to be stored as the last member of the item array X1 in the second AA structure. An address is set in the X1 array to point to the actual disk storage location. Naturally, the statement assumes that two members of the set AA have been created. If there was yet no X1 entry in the structure, the "$+" would signify the creation of the first member. The target is, of course, the position of an item, as it must be in the create process since a buffer is the value. The statement:

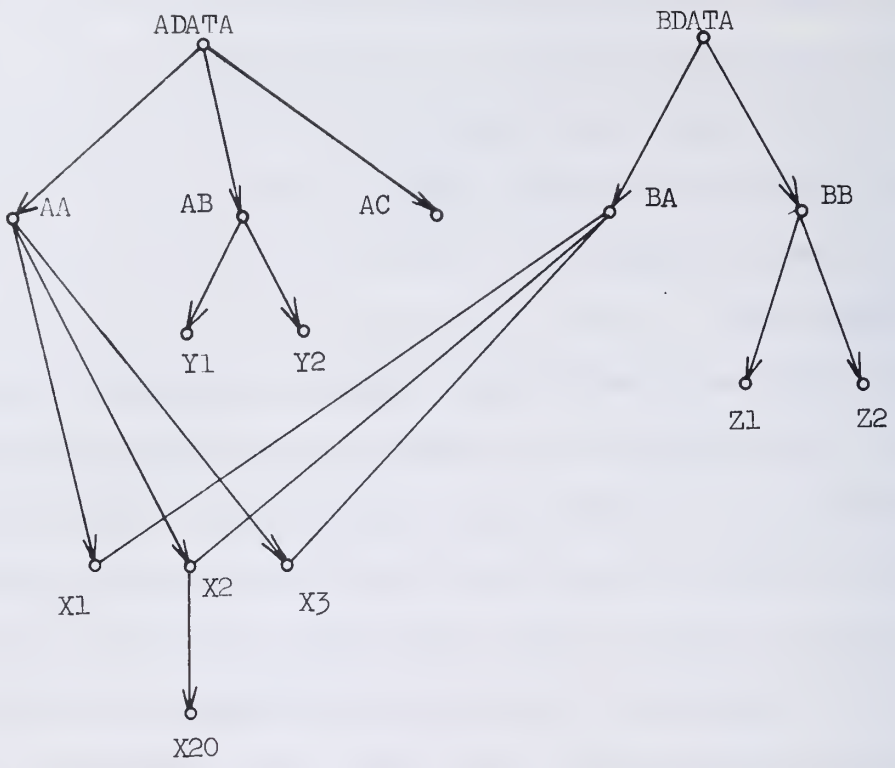


Figure 19. A Graph of an Example Database Construction.

ADATA.AC (3-) ← #(1)

puts the new member between entries two and three in AC. Again, it is assumed that there are at least three items in AC. At the point of creation, number three becomes four, four becomes five, etc. Now consider:

ADATA.AB (2+).Y2 ← #(2)

Although it is an item again being added, this time it is to a structure not yet existing. Therefore, there is no question as to the value of indices below the new level, they must be one. Another example of this is:

ADATA.AA (1-).X2.X20 ← #(1)

A similar function to the create is the write. This is the process of taking data from work areas and overwriting existing data items. For example:

ADATA.AC (2) ← #(2)

Note the absence of any "+" or "-"; these symbols would cause the creation of new data formations. The meaning is that the second work space is to be written onto ADATA.AC (2) which has been previously created. Another write operation might be:

ADATA.AA ($).X1 ($) ← #(2)

Characteristic of the write is the fact that the target is again the location of an item and the value is a work space.

Also a necessary function of data management in its interaction with using programs, is the read. It is the user's only method of retrieving the data which he stores. Naturally, he can only recall items; that is, his work area is unstructured and unsuitable for any data but the simple elements. One such read statement might be:

#(4) ← ADATA.AC (2)

The effect is to take the data from the second element of AC and put it in the fourth work buffer. Another example:

#(1) ← ADATA.AA (2).X1 ($)

The final manipulative operation is that of "relinking." The user can refer to the same physical storage location according to more than one name only by linking the item to other positions in the database after creation. Consider the following:

```
ADATA.AC (1-) ← #(1)
ADATA.AA (2+).X1 ← #(1)
```

If the two operations were performed consecutively, the result would be two separate identical items in disk storage. Therefore, in order for this particular location to be known by both ADATA.AC (1) and ADATA.AA (3).X1, these statements are adequate:

```
ADATA.AC (1-) ← #(1)
ADATA.AA (2+).X1 ← ADATA.AC (1)
```

A similar relinking accomplishes a replacement of existing data, and is much like the write. For example:

```
ADATA.AA (1).X1 ← ADATA.AA (1).X3
```

The difference is that no new structure or item has been created. Structures may also be relinked, as the following illustrates:

```
ADATA.AA (1+) ← BDATA.BA (2)
ADATA.AA (3) ← BDATA.BA (1)
```

The structures on both sides are of the same form and therefore the operation is meaningful. Notice, in fact, that AA and BA belong to the same SET specification. Because of this, resolving that the two sides are structurally equivalent is relatively easy. However, consider this statement:

ADATA.AB (2) ← BDATA.BB (3)

Inspection of the sample DDL shows that each side has the same form, despite the fact that they are not in the same SET statement. More complicated database description, though, might cause expensive overhead of form checking. Moreover, if sets do not represent naturally alike data, then no such relinking (that of structures not of one SET) should be done. This is a problem which must be considered during implementation.

When like forms are defined by the same SET statement, the structure may be defined but once and pointed to by all set names. This creates the easy verification ability which was discussed earlier. A special meaning might be given to the first set name specified in the statement. If horizontal access is to be an important capability, this name might reference a master index of all structures (in the SET). This convention would cause a create of the form <set - 1> ($+) ← <set - i> (last created index), $i \neq 1$ whenever a member of <set - i> was created. Possibly not every set should have such a master index, and an additional syntactic change should be incorporated to specify a special name.

Now consider the example DDL given at the end of the last chapter. Assume that the program set utilizing this database is similar to that of the MIS on PLATO described earlier. After the physician logs on the system, he must choose either to create a patient record, or access an old one. In the former case, he will go on to supply identifying

data, and the program which accepts this information performs the following:

MEDRECS.RECORDS ($+).ID ← #(1)

The operation not only stores the identifying data but also creates a new entry in the RECORDS set. Suppose that the user continues on into the history program. Here several history items may be created and retrieved, such as:

MEDRECS.RECORDS ($).HX ($+) ← #(1)

MEDRECS.RECORDS ($).HX ($+) ← #(2)

#(3) ← MEDRECS.RECORDS ($).HX (1)

The physical examination would call for similar operations. During problem entry, the program might request the following for the specification of a subjective part of a progress note:

MEDRECS.RECORDS ($).SUBREC.PROBLIST ($).PROBNOTES ($+).S ← #(1)

This is, indeed, cumbersome notation. And, in implementation, an easier method of reference is necessary. In addition, each level of structure must access an index list to locate descendants. In light of these things, it seems advantageous to be able to step down level by level keeping indices as they are passed. At the same time, if address vectors were also kept, when steps are made upward to change indices, the addresses are there for accessing other descendants. If the above process is redone using this scheme, the physician first creates the patient record with a new function which "loads" the set. Thus it will be called a LOAD and the first operation will look like:

LOAD MEDRECS.RECORDS ($+)

This establishes the position in the database up to the RECORDS level. Then, to supply the identifying data:

ID ($+) ← #(1)

HX ($+) ← #(2)

#(3) ← HX (1)

Before the user gets to the point of executing the next given example, the following might be done:

LOAD SUBREC.PROBLIST ($+)

This creates a new problem entry and the subjective note may be submitted:

PROGNOTES ($+).S ← #(1)

Now if the physician decides to change to a different problem, the program might execute:

LOAD PROBLIST (2)

All the structures attached to the previously referenced problem are aborted, awaiting operations such as:

PROGNOTES (1).O ← #(2)

At this point, PROGNOTES for problem two are loaded, until something like:

PROBDESC ← #(3)

Then PROBDESC replaces PROGNOTES and O.

At times, it may be necessary to manipulate structures which are not loaded, and yet save all loaded arrays in anticipation of further use. In the above example, after the physician had created the

record, he might have decided to add the record to some family set. The program might initiate:

```
FAMILIES (10).FAMRECS ($+) ← RECORDS ($)
```

This operation would abort the RECORDS structure, and it would have to once again be retrieved for subsequent processing. So, to circumvent this, certain load functions will be done with a save of the array until it is explicitly unloaded. Two new operations then are needed LOAD-SAVE and UNLOAD. And, to again follow the procedure of adding the record to a family:

```
LOAD-SAVE MEDRECS.RECORDS ($+)
FAMILIES (10).FAMRECS ($+) ← RECORDS ($)
UNLOAD FAMILIES
```

The final unload aborts the FAMILIES index. Certainly if too many arrays are saved and not reloaded, an overflow of available array space will occur.

The program set on PLATO which makes up the MIS, directly relates to the structure of the medical record. As the user proceeds deeper into the record, successive loads may be done, allowing each program to work at its own level without excessive overhead. This necessitates greater coordination between program modules. Upon entry to the problem list module the instruction

```
LOAD SUBREC.PROBLIST
```

would be executed under the assumption that the above structure had been loaded.

In order to minimize the loading operations, some items or structures might be designated as being only single copies (not arrays) as an additional deviation from the DDL specification. For instance, the structure SUBREC is only intended to occur once for each record, and therefore it is unpractical to cause an extra load for that level.

Implementation Considerations

To accomplish all of the data manipulation operations, the data manager must be able to internally follow the structure to address actual data on disk. Table organization is important in facilitating the process. The table consists of three basic entities: the name list, the structure list, and descendant pointers. Each entry in the name list is an alphanumeric name and a pointer into the structure list if the name has been entered in a SET or DATABASE statement. Each member of the structure list denotes the type (DATABASE OR SET) and points to a vector of descendants, each of which in turn points into the name table for the name of the descendant. Given any name, it can be determined if that name is a structure by checking for a pointer back into the structure list or for a descendant pointer from the structure list. Likewise, it is possible to check for correct descendant specifications by following the loop from name to structure to descendants to names. Names which are listed in a single SET form a somewhat different entry in the structure list. The first one given will set up pointers as usual, but the others instead of pointing to a distinct descendant

vector, will point to the first name's vector. This will closely connect the names of a SET statement; checks for like structures can be done by comparing pointers to descendant vectors.

Along with the data description table, there must be a table of addresses for the descendants of each entry name defined in a DATABASE statement. These addresses point to disk space allocated as arrays of addresses of further descendants. Each such address vector is as long as there are members in the array, and an entry in the vector is as many addresses as there are descendants to that particular set name. It is this type vector which is loaded during execution. At these times, very large arrays become impossible or impractical to load completely. In this case, the capability must exist for loading the array partially while unknown to the user. Then for access to other portions of the array, another partial load must be done.

In addition to the functions discussed for database modification and retrieval, the using program must have available a system of status flags. With checks on these flags, certain pertinent information can be learned for processing, including error signals and the state of the database. Present indices of loaded arrays, range of possible values, and security status should also be acquired through the system.

IV. SUMMARY

Health care is indeed a large and growing industry. It places a great demand on our national resources, be they human, technological, or otherwise. If the nation is to fulfill its commitments for comprehensive medical care, it must responsibly and efficiently utilize all such resources. Over recent decades, the computer has come into view as an aid to a spectrum of applications. Its computational powers were apparent at its birth, many years ago. Since that time, the computer has evolved through advances in electronics, and systems have been conceived for the processing of more than just numbers--information. The very generality of the concept of dealing with information, storing, translating, and retrieving it, lent usefulness to a nearly inexhaustible list of disciplines. And now, many uses are being realized, others investigated. Among the latter is the medical information system. Many years ago, the medical record could have been stored on the computer. However, in its nonstructured form, there was no advantage over paper methods, and was, in fact, costlier. Research done at the University of Illinois has demonstrated that the record keeping process can be cost-effective on a PLATO-like system. In addition, recent advances in record organization have improved the data gathering and retrieval process in patient care. With the structured problem-oriented medical record, the physician's clerical functions become systematized, leading to more complete patient data. The hierarchical nature of the data is natural for computer programming and presentation. The PLATO system,

in particular, can graphically display portions of data, further illustrating advantages of automation. Through physical access to the terminal, the physician may acquire elements of an entire database, bypassing tedious manual filing. And, if properly used, facilities for touch input can reduce the time that the user requires for entering various parts of the record, and possibly create a more acceptable atmosphere for interaction at a terminal. This man-machine interface is an important subject under investigation particularly relevant to medical record computerization. Physician acceptance of any system is an undeniable criterion during development. It is just this fact which requires health care personnel participation in the research. Input from physicians and nurses at operational testing sites can be invaluable.

The means and environment exist for an adequate medical information system. The great advances in medical record keeping have inspired research for such a system. Establishment of the initial database for the patient marks the creation of the record. With the knowledge that the physician acquires from laboratory tests, a patient history, a physical examination, and the patient interview, he can formulate a list of problems upon which he must take action. It has always been an important task of the physician to discern what observations may be symptomatic and what may be diagnosed from symptoms. The problem-oriented medical record only proposes a method of organization. Yet, it may be sufficient, in itself, to improve the quality of health care delivery. With an increasing number of specialists, a clear, precise record, more easily understood by a multitude of viewers, is definitely

in need. The problem list provides an index into the record, and a great deal can be learned from a cursory glance. The physician takes his knowledge of the patient, and uses his training and experience to initiate a plan for treatment or tests. Certainly, more data is generated, and thus, the cycle begins--the evolution of the record by the continued addition of progress notes.

The structure of the patient record leads us to believe that we can more formally organize the medical database to best serve the corps of interested users. It is clearly evident that not everyone wishes to deal with every facet of the record, or even the record itself, as it is thought of conventionally. Nor is it wise to allow semirestricted access to confidential information. Therefore, the criteria for database organization must be capabilities of flexible modes of access and security of all levels of structure. Naturally, this includes methods of forming the problem-oriented record. The MIS project partially meets the requirements. Vertical accessibility is possible for the physician, nurse, and clerical personnel, who desire data associated with an individual. Programming allows pieces of the record to be related together, but no internal structure is maintained beyond a separation of identifying and medical data. Security is leveled at the two areas distinctly and by user group.

It should be the function of the database management system to impose a structure on the data, thereby leaving items for user-defined data only. Also, all operations involving the basic management of

the disk memory should be transparent to the user. A proposal has been made for describing the medical database. All segments of data are stored according to the nature of their content by a particular name. According to the descendant relationships which can be defined, a referencing convention has been described. In addition, structures may be iterated for sets of like data. The system allows the application program to generally handle data which is relevant to its purposes and not need to concern itself with the more basic maintenance functions. The data description language makes it possible to specify what the database will look like to a system which is designed to operated on any structure described thusly.

The using programs also have available a set of transactions which may be made with elements of the database. These actually constitute the interface between programs and the data management system. The functions allow for the storage and retrieval of user-accumulated data from and to work buffer areas.

Security, another necessary quality of the system, may be required at any hierarchical level of data. The method of individuals belonging to groups, as implemented in the MIS project, appears to be a viable means of categorizing potential users for security classification.

The health data collection process is one for which the physician and related personnel are largely responsible; therefore, patient data generally originates in the medical record. Proper formation of

this documentation betters our chances for complete and responsible treatment. If computers are to be instrumental in bringing about the utilization of technological tools by the medical profession, then the correct weight of efficient record keeping must be merged with a sufficient amount of natural organizational database characteristics, to produce a smooth, practical information system which has the approval of those who directly interface with it.

LIST OF REFERENCES

1. Weed, H. R., "Biomedical Engineering in Health-Care Delivery," IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Vol. BME-20, No. 3, pp. 161-164, May 1973.

2. Ball, M. J., "Medical Data Processing in the United States," HOSPITAL FINANCIAL MANAGEMENT, Vol. 28, No. 1, pp. 10-30, January 1974.

3. Williams, B. T., "Source Interactive Medical Information System." A grant application for the Regional Health Resource Center. June 1974.

4. Grabieli, E. R., "Computerization of Clinical Records," Vol. 1, Ch. 3, pp. 1-5, Grune & Stratton, Inc., 1970.

5. Weed, L. L., "Medical Records, Medical Education, and Patient Care," pp. 3-123, The Press of Case Western Reserve University, 1971.

6. Sandlow, L. J. and P. G. Bashook, "Problem Oriented Medical Records (Self-Instruction Handbook)," pp. 2-30, Michael Reese Medical Center, 1974.

7. Harmon, L. D., "Some Problems and Priorities in Health-Care Technology," IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Vol. BME-20, No. 3, pp. 216-220, May 1973.

8. Saltzer, J. H., "Protection and the Control of Information Sharing in Multics," COMMUNICATIONS OF THE ACM, Vol. 17, No. 7, pp. 388-402, July 1974.

APPENDIX

The following documentation has been copied from specifications for the Patient Record Manager which reside on-line on PLATO. The data management system of the medical information system is presented in detail. The figures are reproductions of their PLATO screen appearance.

The Patient Record Manager
Version 2.3
System Specifications

by

David Mueller
Robert Rush
David Fumento

June 26, 1975

14.14.25.

Part I. General Overview

1. Introduction and Goals

The Patient Record Manager (PRM) is the component of the Medical Information System (MIS) which is responsible for the storage and management of patient records. Specifically, the PRM 1) creates and maintains the patient record on permanent storage, 2) inserts data into the record as instructed by the input components of the MIS, and 3) retrieves information from the record as instructed by the output components of the MIS. Access is vertical by patient record rather than horizontal by data item.

The PRM must meet sets of criteria imposed by the nature of the medical record, by PLATO, and by other components of the MIS. First, the PRM should be able to maintain records which can expand to any size and maintain their security, confidentiality, and integrity. The PRM is responsible for storing and retrieving all data in the patient record, but it is not responsible for maintaining the relationships between the various data items. It should give as rapid a response as possible and should allow for record reorganization based on use patterns.

The PRM must operate within the constraints imposed by PLATO. It should require as little duplicated code as possible. Further, it should make the best use of the algorithms used by PLATO to allocate system resources such as CPU, ECS, and the disk channel in order to achieve adequate response. Within the constraints imposed by the MIS,

the PRM must use no more than 1,000 words of storage for i/o buffers and work and status variables, and should use no more than 6,000 words of combined common and condensed binary code.

Since the structure and access patterns for the medical record are presently a topic of research, the PRM should be flexible enough to allow for changing approaches. It is also desirable to be able to insert with relative ease routines to monitor the pattern of use of the record. Finally, straight-forward, simple procedures for accessing the PRM are desirable.

This report describes the components of the PRM which attempt to meet these requirements. First, a general overview of the PRM is given, followed by detailed descriptions of each component.

2. Record Security

We believe that at least two record security problems must be handled. The first is the use of the MIS to gain unauthorized access to the records. To combat this, the PRM will provide site checks in the router to restrict the lesson to specified terminals. In addition, passwords will be required to enter the system. Finally, the PRM provides "by-record" access control so that otherwise valid users may be denied access to certain records.

Each user is assigned a twelve bit user number and a twelve bit user group number. Each record has associated with it a table containing an entry for each user group recognized by the system. Each entry gives

the level of access for that group to that record, as described in section 7.1.1. This method allows greater ease in creating and changing access rights, and results in a shorter access control list.

The second security problem involves bypassing the patient record system entirely and reading the data files directly. PLATO provides a minimal amount of protection in the form of code words attached to the file. This is sufficient to prevent casual browsing through the files, but is an inadequate defense against determined attempts to read the files. Not only may the code words be broken, but the code word system may be bypassed entirely as is done when system changes are made to lessons.

Therefore, in addition to placing inspect and change codes on all lessons and data files, the records themselves might be enciphered. Each record would have associated with it a cipher key, which must be used to decipher the record. The cipher keys would be kept in the Master Patient Directory, which is itself enciphered. The Master Directory's cipher key would be changed and the Directory reciphered as often as is necessary to protect the patient cipher keys.

This level of security may be more than the users are willing to pay for, and, since most of the data is encoded by dictionaries, a trivial level of enciphering is already employed. Therefore, the initial releases of version II will not employ ciphering and ciphering will possibly be introduced in later releases.

3. Flow of Control

The PRM has been divided into four major components: 1) the Sign-On Component (SOC), 2) the Master Directory Component (MDC), 3) the Item Management Component (IMC), and 4) the User Service Component (USC). These are described in greater detail in section 5.1. Briefly, however, the SOC is responsible for identifying and validating terminal users and the MDC for allowing the terminal user to create new records and access old ones. The IMC and the USC as a package provide data storage and retrieval functions to the applications lessons, the IMC executing the functions and the USC providing the interface.

The way in which the above components fit together is important to understanding the overall structure of the PRM. What follows is an overview of the sequence of operations carried out by each component, with descriptions of what the terminal user sees and what the applications programmer must do.

3.1 General Overview (see Figure 1)

The terminal user will see the following sequence of operations. Using the SOC, he will identify himself to the MIS. After verifying that he is a valid user, the SOC will exit to the MDC to either access a patient record or create a new one. The user then has the option of entering or reviewing medical data in the record using the IMC, the USC, and the applications programs. At any time he may return to the MDC to request another record.

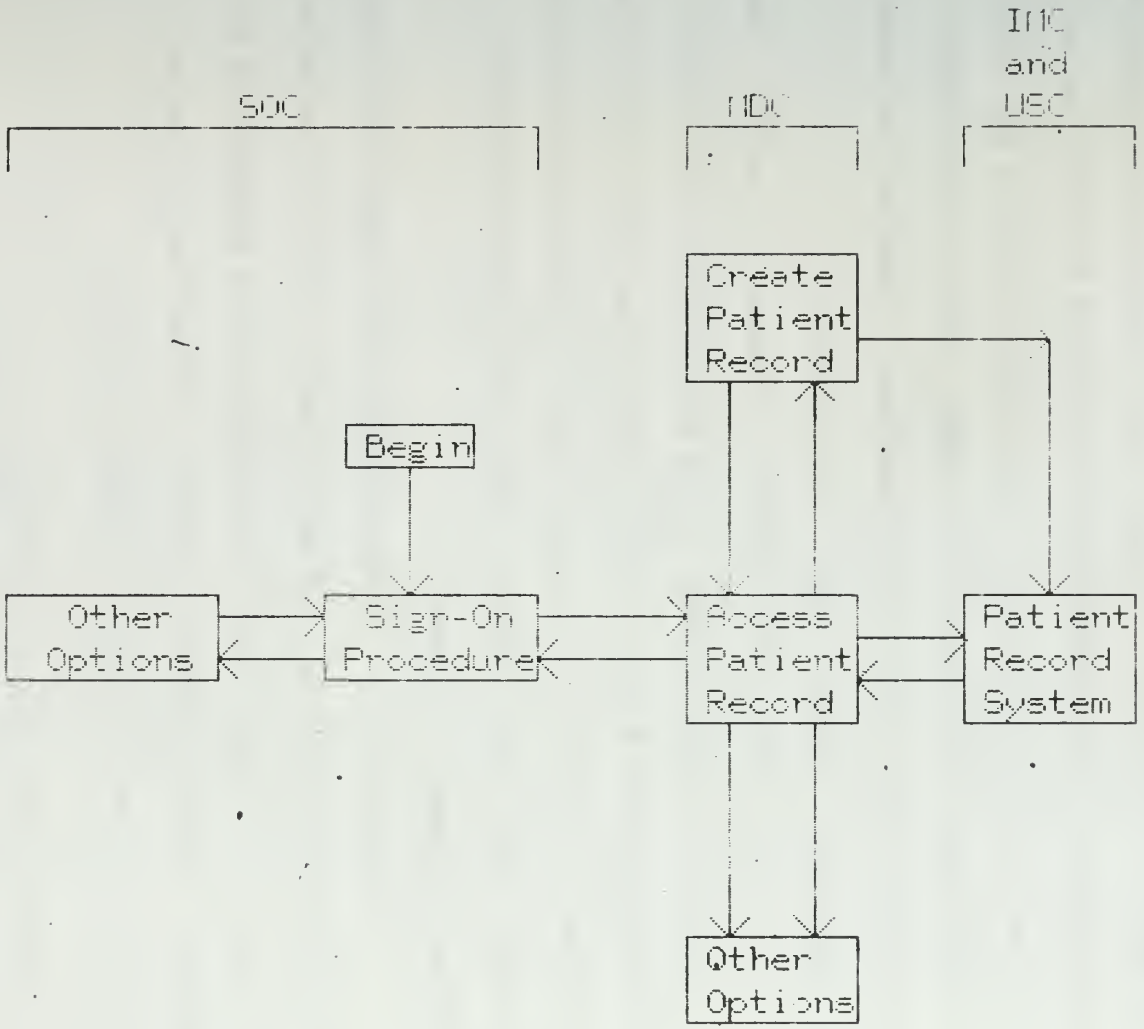

SOC
MDC
INC and USC
Create Patient Record
Begin
Other Options
Sign-On Procedure
Access Patient Record
Patient Record System
Other Options

Figure 1

3.2. User Sign-On (see Figure 2)

The identification and validation of terminal users is based on the following structure. The users are divided into user groups not mutually exclusive. All members of a single group have the same access to the same set of patient records. Thus, each user has a user number and a group number associated with his name. Access to the records is based on group number while trace information is based on user number. This is described in greater detail in section 4.1.

The user first sees a copywrite statement for the MIS. Then, if his terminal is not designated to the auto-sign-on, he is asked to select his group from a list of groups and his name from a list of users in his group. Selection may be made either by touch or by keyboard entry. If only one choice is possible on either of these pages, the choice is assumed and the page is skipped.

If the terminal is designated to be auto-sign-on, the prespecified selections are made and the user is asked to fill in the unspecified selections. In any case, the user must enter a password and is then taken to the MDC to select a record. The user may also go to a page listing options available to unidentified users.

3.3. Opening Patient Records (see Figure 3)

After entering a valid password in the SOC, the user is asked for the PIN of the record he desires. Based on this, the Master Directory is searched. Three outcomes are possible: 1) the record may not

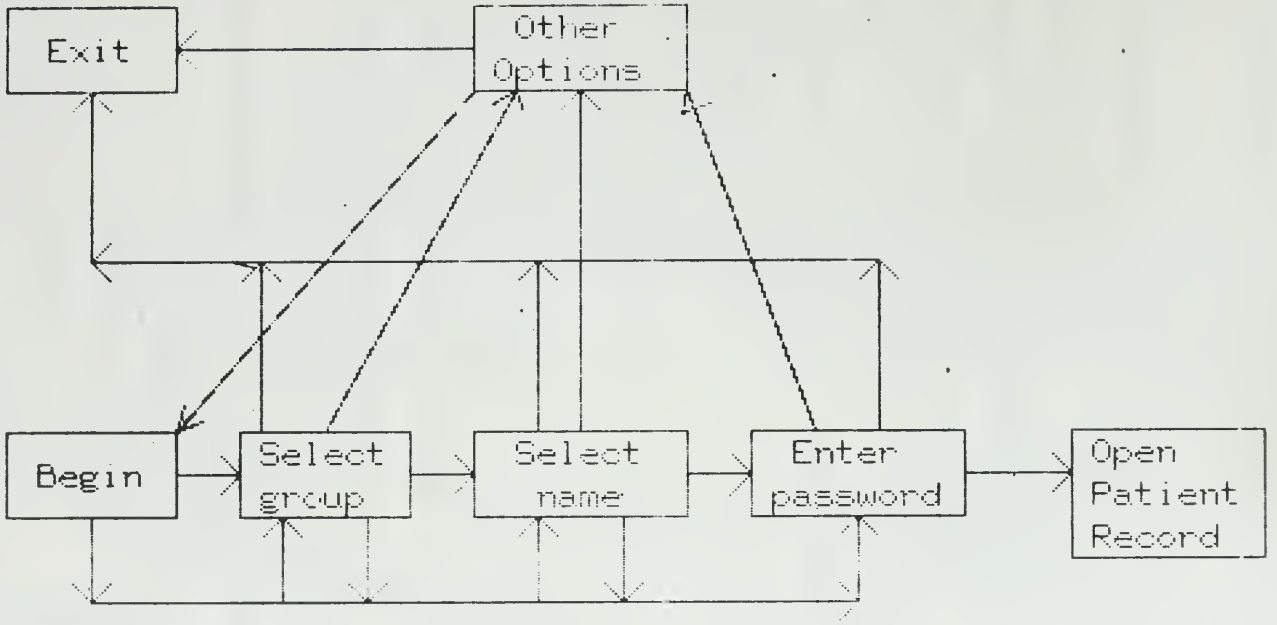

Exit
Other Options
Begin
Select group
Select name
Enter password
Open Patient Record

Figure 2

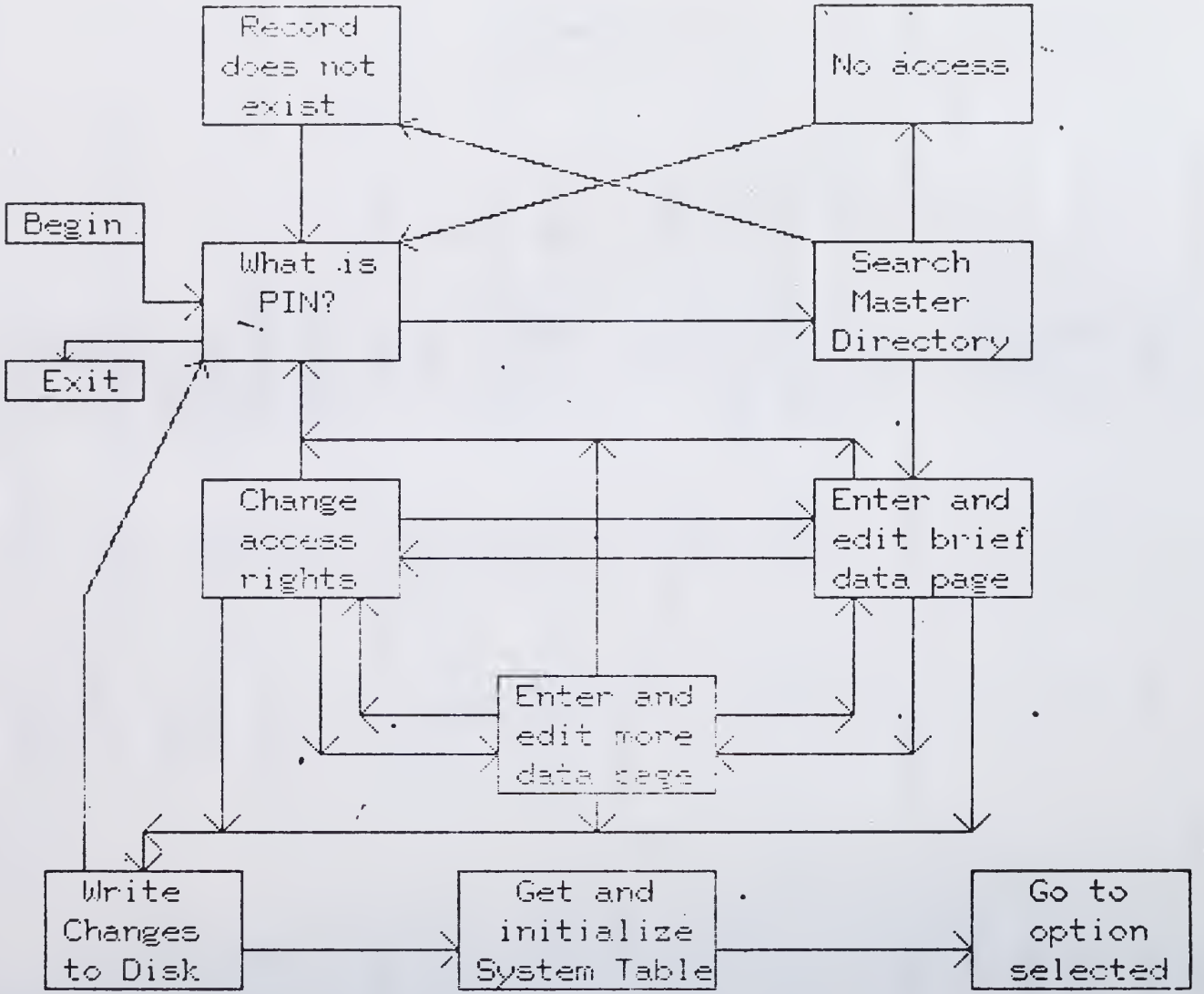

Record does not exist
No access
Begin
What is PIN?
Search Master Directory
Exit
Change access rights
Enter and edit brief data page
Enter and edit more data page
Write Changes to Disk
Get and initialize System Table
Go to option selected

Figure 3

exist, 2) the record may exist but the user does not have access, or 3) the record may exist and the user does have access. In the first two cases, the user is informed of the situation and allowed to re-enter the PIN.

If the record does not exist and the user does have access, then the brief data page for the record is displayed. At this point the user has the options of 1) editing the data page, 2) editing more data, 3) changing access rights, 4) proceeding with the open, or 5) exiting all of these subject to his own access rights. If he edits the data page, the results may be stored by proceeding or ignored by exiting. If the user desires to proceed system control blocks are obtained for him and he proceeds into the record.

3.4. Create Record (see Figure 4)

After selecting the create record option on the PIN request page the user's group is established as the owner of the record. This sets up a predefined access control list for the record.

The user is then taken to the brief data page. The patient name is required at this point, and other data may be entered as desired. The user may also branch to a page which allows him to alter the access control list, or to a page listing more ID. On each of the three data pages he may abort the entire procedure.

Upon completion of either the brief data page, the more data page, or the access control page, the user may order the record created.

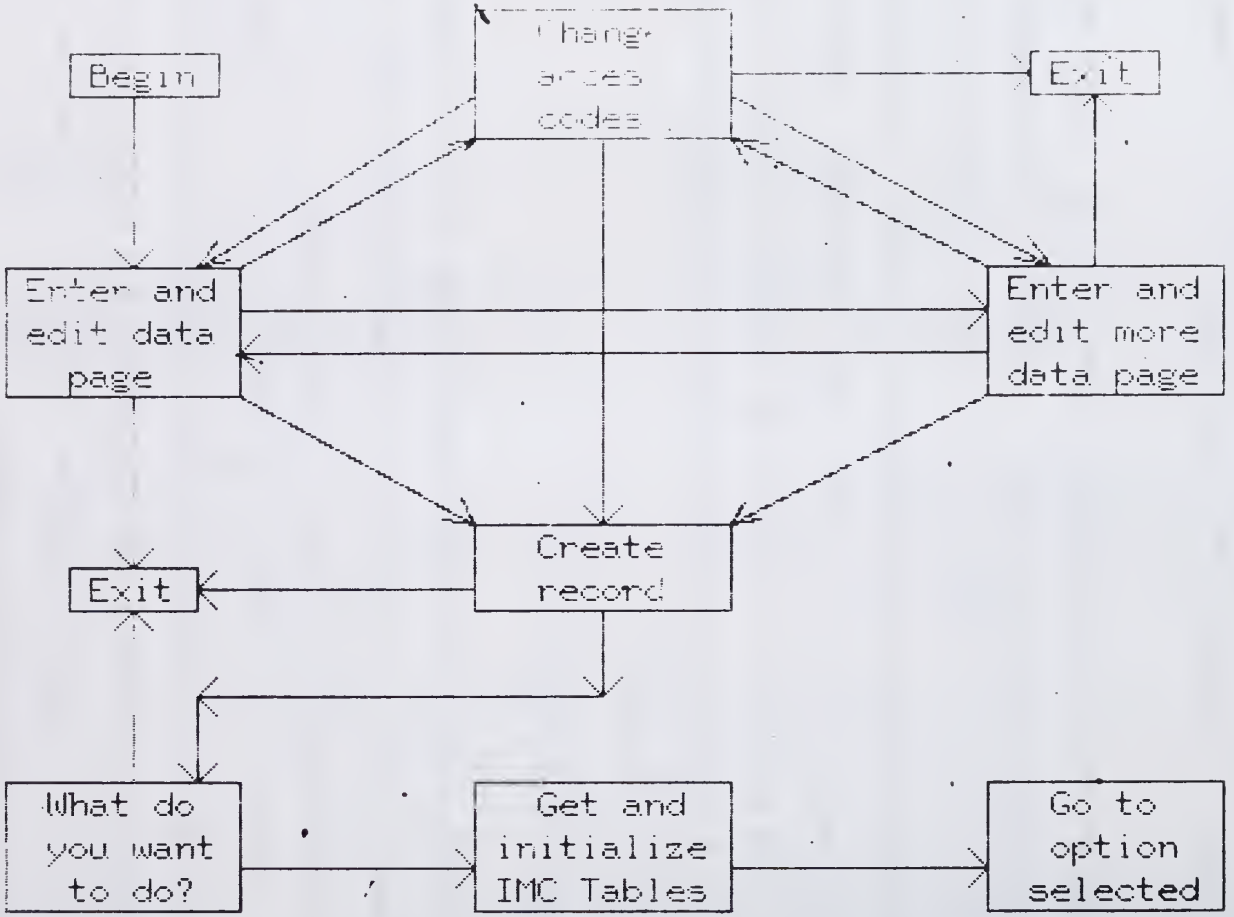

Begin
Change area codes
Exit
Enter and edit data page
Enter and edit more data page
Create record
Exit
What do you want to do?
Get and initialize IMC Tables
Go to option selected

Figure 4

At this point a Patient Index Number (PIN) is assigned to the record, and the Master Directory and Patient Directory entries are created and initialized. The user may now elect to enter medical data or exit. If the choice is to open the record, system control bocks are obtained and initialized and the user is sent into the record.

3.5. The Item Manager and User Service Components (see Figure 5)

The Item Manager Component is called by the applications program using the facilities of the User Service Component. This component consists of an interrupt routine and a collection of interface routines, one for each function provided by the Item Manager. The applications program calls an interface routine for the particular function it desires. This routine, after setting function dependent variables, calls the interrupt routine, which sets the interrupt variables and signals that an interrupt is present.

The IMC consists of a driver and a collection of routines for performing the IMC functions. Upon receiving the interrupt signal, the driver determines which user is requesting service and what service is desired. It then calls the appropriate routine to actually do the work and, upon its completion, signals completion to the interrupt routine in the USC.

The interrupt routine, which has been waiting for this signal, now transfers data from IMC buffer to the applications program's data space and returns control to the calling routine. Should a disk error

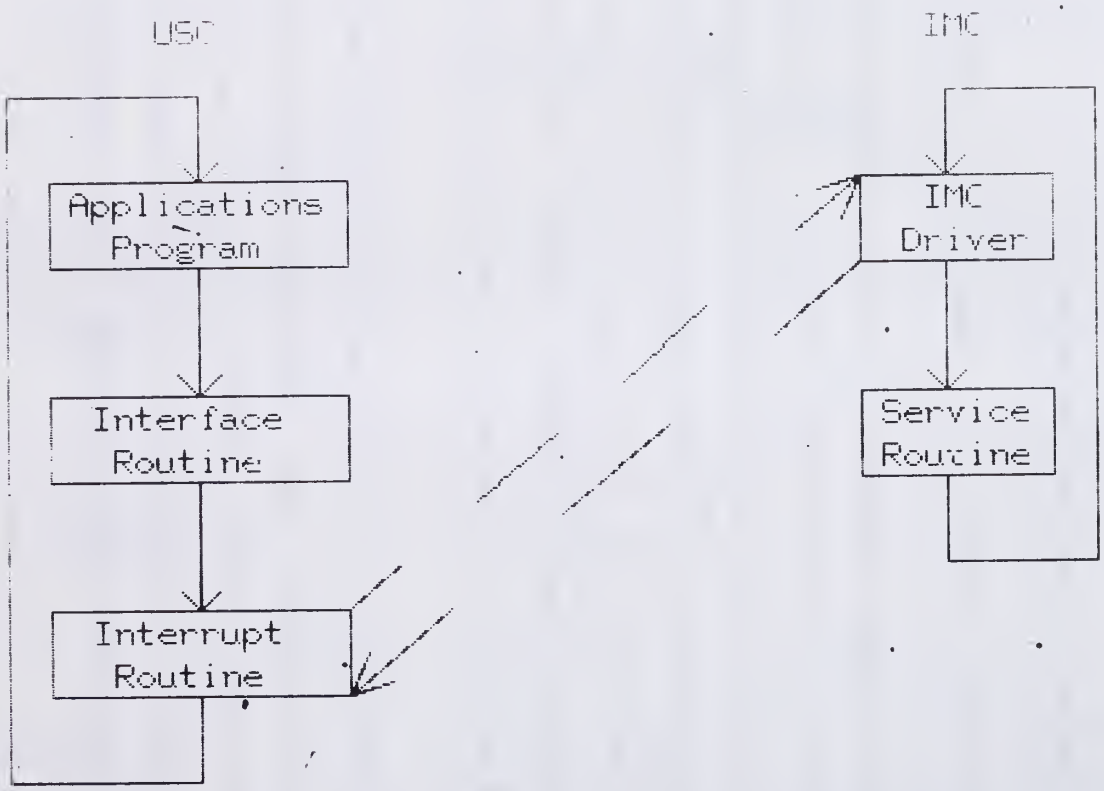

USC
IMC
Applications Program
Interface Routine
Interrupt Routine
IMC Driver
Service Routine

Figure 5

occur in the IMC, it will be noted in the IMC, and the applications lesson will be aborted.

This scheme is only the current implementation and might need to be altered by a growth in the number of users or changes to Tutor. In particular, one of the alternatives envisioned for accomplishing the function of an interrupt is the use of a jumpout command. Therefore, the IMC should be called only from a main unit, since the jumpout would destroy the base pointer.

4. The Patient Record

4.1. PRM Viewpoint

The PRM views the patient record as consisting of two major sections, a set of identifying and control data and a set of medical data items. The identifying data consists of data which might identify the patient record, such as the patient's name, Social Security number, next of kin, and so forth. The control data consists of systems data used to control access to the record. The identifying and control data are kept separate from the patient's medical data and can be altered only by authorized personnel.

The PRM views the patient data as consisting of a set of named, unrelated data items. Data items may vary from 4 to 319 60 bit words, and may contain such information as a problem name, a prescription, a section of a physical exam, a patient history, laboratory data or summary information. The PRM stores and retrieves such data items by name, but

does not establish or maintain any relationships between items, this being the responsibility of the input, process and output components of the MIS.

There are three types of data items: 1) permanent, 2) temporary, and 3) rewriteable. Permanent items contain data which cannot be altered. Such items are stored at the time of creation and cannot be rewritten after that. Temporary items may contain intermediate checkpointed results stored in order to facilitate restarting should the system "crash." Such items may be rewritten and eventually deleted, and are of a fixed 319 word size. Rewriteable items contain summary or other data in the record and may be updated, but not deleted. Rewriteable items may be extended up to the maximum item size of 319 words.

4.2. Internal Structure of the Patient Record

Internally, the patient record consists of five major sections: 1) the system control data, 2) the identifying data, 3) the block directory, 4) the item list, and 5) the patient data. The system control data and the identifying data are kept in the Master Directory, while the block list and item list are kept with the patient data.

The system control data and the identifying data have been described generally in section 4.1. Their content will be discussed in detail in section 7.1. The block list is a translation table relating the patient's virtual data space to PLATO's physical disk space. This is a paged list with each page capable of referencing 24,960 words (78

blocks) of patient data. Also included as part of the information stored in the block list are the free space list pointers. Free space results when physical blocks are allocated to the record but the entire space is not assigned to items, or when items are extended. One list is kept for each potential size of free space block, and the length of each list is kept to facilitate garbage collection.

The item list relates item names to locations in the patient's virtual data space. It contains, for each item, an item information word (IIW) which contains the item name, type, size, and location. The first word of each item then contains the trace information for the item, such as the creator and date and time created. This method approximately doubles the number of item names which may be kept in a single list and, since the trace information is altered only at the time the item is written, no extra disk accesses are required. There is room for a total of 297 items in the item list.

5. General Structure of the PRM

5.1. Lesson Structure

The assignment of components to physical lesson spaces and the general conventions for calling the routines follow roughly the same pattern as the logical structure. The Sign-On and Master Directory Components run as stand-alone modules, with their own input and display routines. These components are accessed by jumps to the appropriate entry points. The User Service Component is, by its very nature, a shared

routine whose code is via "use" commands. Routines in this component are called by executing "do" statements referencing the appropriate entry point.

The Item Manager Component presents special problems. Like the service component, its code consists of routines which are shared by other elements of the MIS. However, the ECS required by the IMC code is far too large to be duplicated in the large number of lessons required in the MIS. Therefore, the IMC must reside in a single lesson, and its code be accessed with some mechanism other than a "do" statement.

Two alternatives are available. The first is to use a "jumpout" to access the IMC. This scheme offers the advantage of using the PLATO resource allocation algorithms to allocate the CPU time and disk accesses, thus penalizing only those users who to use more than their fair share of the resources. However, this scheme does require the use of an oversized storage allocation for the IMC workspace, thus using potentially large amounts of ECS. Furthermore, if restrictions are placed on the frequency with which jumpouts may be made, the whole system performance may deteriorate.

The second alternative is to run the IMC as a coroutine accessed by semaphores or software interrupts. This would require the use of a router to contain the IMC code, in order to obtain a communication area independent of lesson common. This scheme offers the advantage of making the gathering and display of system statistics and the monitoring of system performance easier, and does not require the use of large amounts of storage.

The scheme does limit the number of active users to a maximum of 10 users for an 8,050-word router common. Moreover, abuse of the system resources by one user would degrade the entire system performance, since all the disk accesses for the patient data and CPU resources are being used by one PLATO user.

The latter alternative has been chosen at this time, largely because of its greater flexibility. Admittedly, it is a temporary solution which will become inadequate as the number of users increases, but will allow the gathering of statistics on use patterns and the design of an appropriate calling mechanism. Every attempt will be made to make the actual mechanism for calling the IMC transparent to the user lessons, so that changes will not affect existing users.

5.2. Multiple Function Accesses to the IMC

A multiple function access capability for the IMC would allow an applications programmer to construct a sequence of IMC to execute the entire sequence. This capability would have the advantage of alleviating any bottleneck involving the interrupt scheme. This has not been implemented in version 2.0 for the following reasons:

1. The cost associated with the actual transfer of control to the IMC would be minimal if a hardware or PLATO systems software interrupt is used. The current scheme using a software interrupt averages from 100 to 200 msecs to complete one direction of the interrupt.

2. It is not at all certain that the interrupt mechanism will be the bottleneck; the disk channel may well be. Further, the cost of

disk accesses is independent of the number of accesses performed. The IMC can read blocks no faster than PLATO allows, regardless of how the interrupts are done. The last disk block read by a particular user is held in a buffer. Thus, the record has been repacked to increase access rates, groups of item accesses will not require additional disk accesses. This will require an intelligent choice of item names, since the repacking routines will probably use these names as the basis for repacking.

3. In the short term, we can create functions consisting of sequences of the basic IMC functions. These could be created for often-used sequences operating on a single item, giving the programmer a limited multiple function capability on a single item.

4. A reasonable multiple item access may be implemented by adding an "execute" function to the USC. The interface routines would then simply add the parameters to a list until an execute function was received, after which the interrupt would be performed. However, greater control would have to be exercised over the requests to keep one user from monopolizing the IMC. Furthermore, the present scheme of transferring data and parameters would have to be modified considerably. Finally, conventions on error recovery would have to be established. In other words, a large amount of design and programming time would have to be invested to alleviate a bottleneck here when the real bottleneck may well be the disk channel.

5.3. Evaluation

Figure 6 gives the number of disk accesses expected for each

Operation	n=Number of Accesses Required
Create Patient Record	n=4
Modify Access Rights	n=2
Open Patient Record	n=2
Close Patient Record	n=Ø
Attach Item	n=Ø
Detach Item	n=Ø
Create Item	2≤n≤3
Delete Item	n=1
Extend Item	3≤n≤5
Get Item	n≤1
Put Item	1≤n≤2
Replace Item	1≤n≤2

Figure 6

type of service offered by the Record Manager. We assume that these operations can be divided into a class of common operations and a class of much more infrequent operations. The more common operations include Get Item, Put Item, Lock Item, Unlock Item, and Create Item, with the last being the operation least frequently used. The infrequent class includes the rest of the operations in Figure 6. This gives a rough average of one disk access per call.

Assuming that PLATO allows an average of one disk access every 15 seconds to each individual terminal, the IMC would be capable of handling an average of one disk access or call every 15 seconds for all users. Thus, if four users were on, they would be limited to one call each minute. Three users would be limited to one call every 45 seconds, two users to one call every 30 seconds, and one user to one call every 15 seconds, on the average.

The figures given here are estimates only, so we intend to measure several parameters in order to make any changes necessary to ensure adequate service. At present, we plan to collect means and standard deviations on the number of disk accesses per call to the IMC, the number of disk accesses per Get Item, the number of disk accesses per unit time in the IMC, and the CPU usage per unit time. Other parameters may be measured as needed.

Part II. Component Descriptions

6. The Sign-On Component

6.1. User-Group Structure

As described in section 3.2, the terminal users are divided into groups. These groups are not mutually exclusive, and may consist of as few as one user. Each group is assigned a number when it is entered into the system tables. This number is associated with the user when he signs on, and is used by the PRM to determine the user's access level to a record.

Each user is also assigned a number when his name is entered into the system tables. There is only one number assigned to a user, regardless of what group(s) he is in. This number is associated with the user at the time he signs in, and is used by the PRM to provide trace information.

This structure provides several advantages. Since access rights are based on groups of users rather than individuals, initializing and changing access codes is fast and easy. A great degree of flexibility is maintained, however, since a group may consist of a single user. Thus, if access control is desired for individual users, these users may be defined as single user groups. Furthermore, since a user may be in different groups, he may be granted different access levels to different sets of records.

This structure is implemented as a set of three tables. One, called the user table (see Figure 7), contains the user names, user numbers, and user passwords.

The second, called the group table (see Figure 8), contains the group names, group numbers, and group templates. This last field is a template used by the PRM to initialize record access lists; if group i is the owner of the record, then the template for group i will be used as the initial access list for the record.

The last table defines the relationship between users and groups (see Figure 9). If user number i is a member of group j, then one of the entries in the relation table will be (i,j).

While the group and user tables are unordered, the relation table is ordered by group number. This ordering makes it easy to access all the members of a particular group, which is the present scheme in the SOC. However, the same structure could be used fairly easily to obtain all groups containing a particular user, if a change in the SOC is desired.

Finally, to provide an auto-sign-on facility, an auto-sign-on table is kept (see Figure 10). This table relates station numbers to user and group numbers. An entry is made for each station which has an auto-sign-on capability. The group number, or the user number or both numbers may be associated with the station; if one is omitted, it must be filled in at the time of sign-on.

Each of these four tables is implemented as a set of parallel segmented vectors, one vector for each field. The same index is used to reference each vector, thus giving a complete entity.

180 Bits	12 Bits	60 Bits
User Name	User Number	User Password
User Name	User Number	User Password
User Name	User Number	User Password
User Name	User Number	User Password

Figure 7

240 Bits	12 Bits	60 Bits
Group Name	Group Number	Group Template.
Group Name	Group Number	Group Template
Group Name	Group Number	Group Template
Group Name	Group Number	Group Template

Figure 8

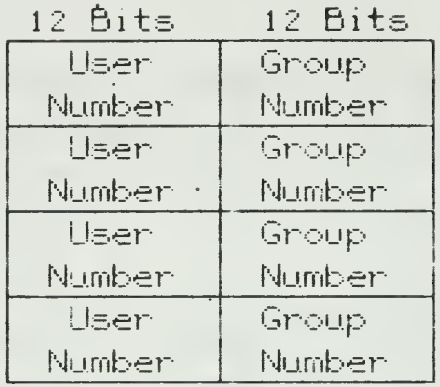

12 Bits
12 Bits
User Number
Group Number
User Number
Group Number
User Number
Group Number
User Number
Group Number

Figure 9

12 Bits	12 Bits	12 Bits
Station Number	Group Number	User Number
Station Number	Group Number	User Number
Station Number	Group Number	User Number
Station Number	Group Number	User Number

Figure 10

6.2. The Sign-On Sequence (see Figure 2)

This sequence was described briefly in section 3.2.

After making preliminary checks and loading the system charset, the lesson presents the user with a copywrite page (see Figure 11). If the terminal is designated as an auto-sign-on terminal, any of the pre-specified data on group and user numbers will be filled in and the corresponding frames skipped.

The next frame presented to the user is a list of groups (see Figure 12). The user is asked to select his group and is presented with a list of users in that group (see Figure 13). On both frames the user is asked to touch the box corresponding to his selection (the keyset is also active). Either of these frames is skipped if only one choice is available.

The user is then asked to select a password (see Figure 14) if none has been selected. This is required of both normal and auto-sign-on terminals. Entry of the password takes place via the keyset, and the user has the option of changing his password (after entering it). Upon validation, the user is taken to the PIN request page in the MDC.

The user may also choose to see other options (see Figure 15). This presents a list of options available to the unvalidated user. Selecting on takes the user to the corresponding lesson.

7. The Master Directory Component

7.1 The Master Directory

Associated with each patient record is a set of identifying

MEDICAL
INFORMATION
SYSTEM

COPYRIGHT REGIONAL HEALTH RESOURCE CENTER 1974

THIS WORK SUPPORTED BY CONTRACT OG-24 OF THE
ILLINOIS REGIONAL MEDICAL PROGRAM

DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE

Figure 11

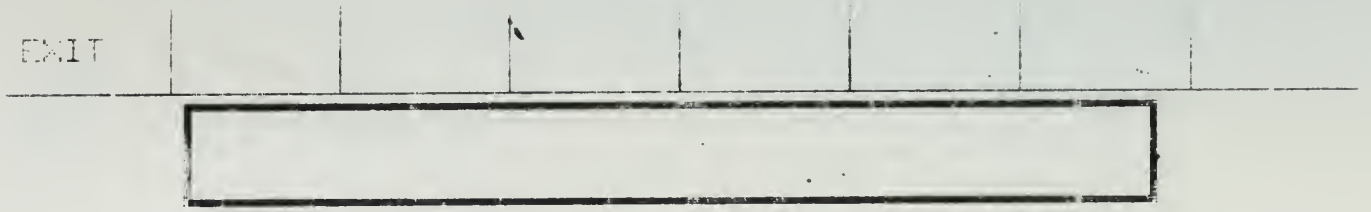

EXIT
Touch the box by your group.
a. D. Schultz
b. Danville Family Practice Center
c. D. R. Hamilton
d. McKinley Health Center
e. Regional Health Resource Center
SOC1
CONT-INUE
Options List

Figure 12

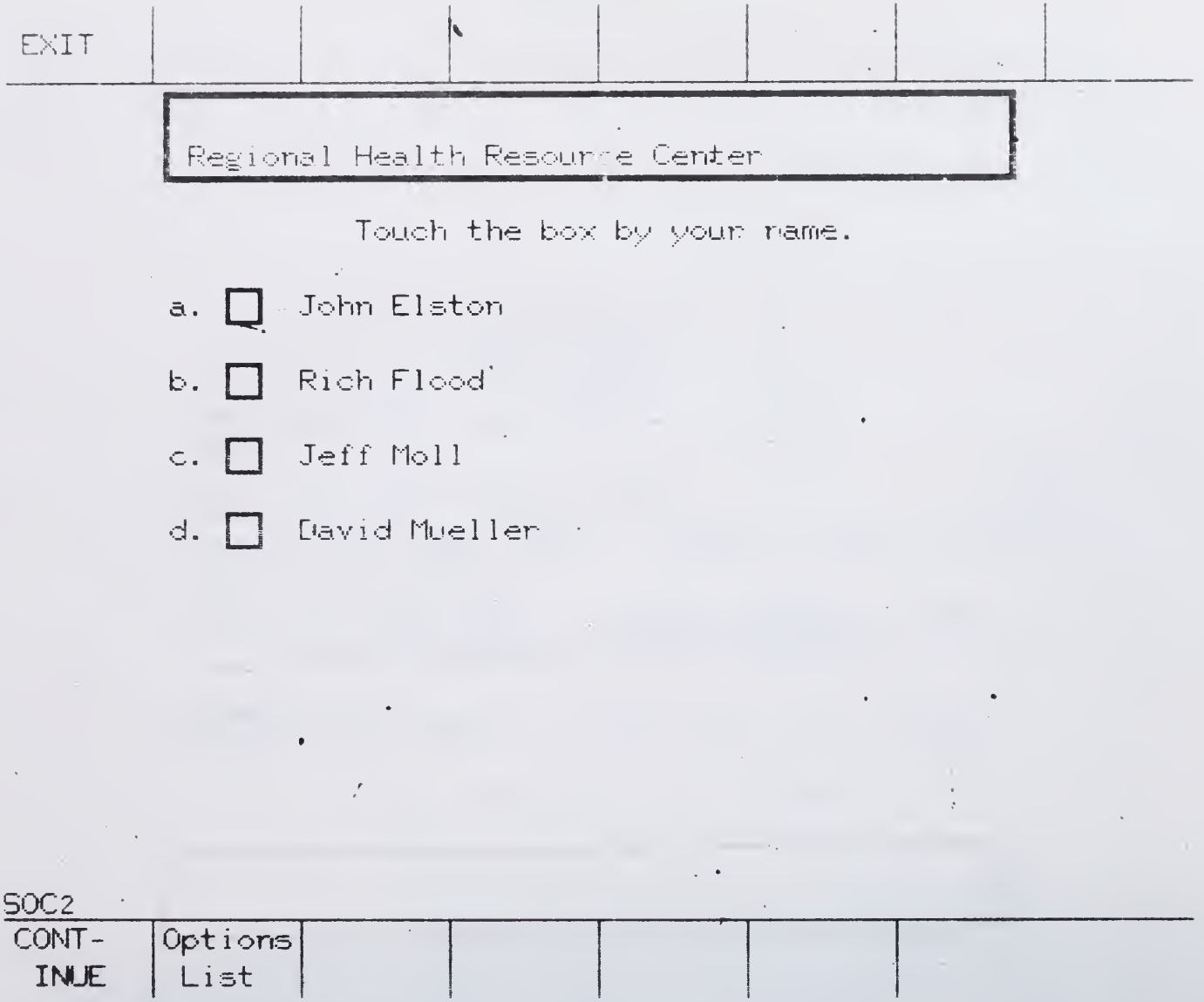

EXIT
Regional Health Resource Center
Touch the box by your name.
a. John Elston
b. Rich Flood
c. Jeff Moll
d. David Mueller
SOC2
CONT-INUE
Options List

Figure 13

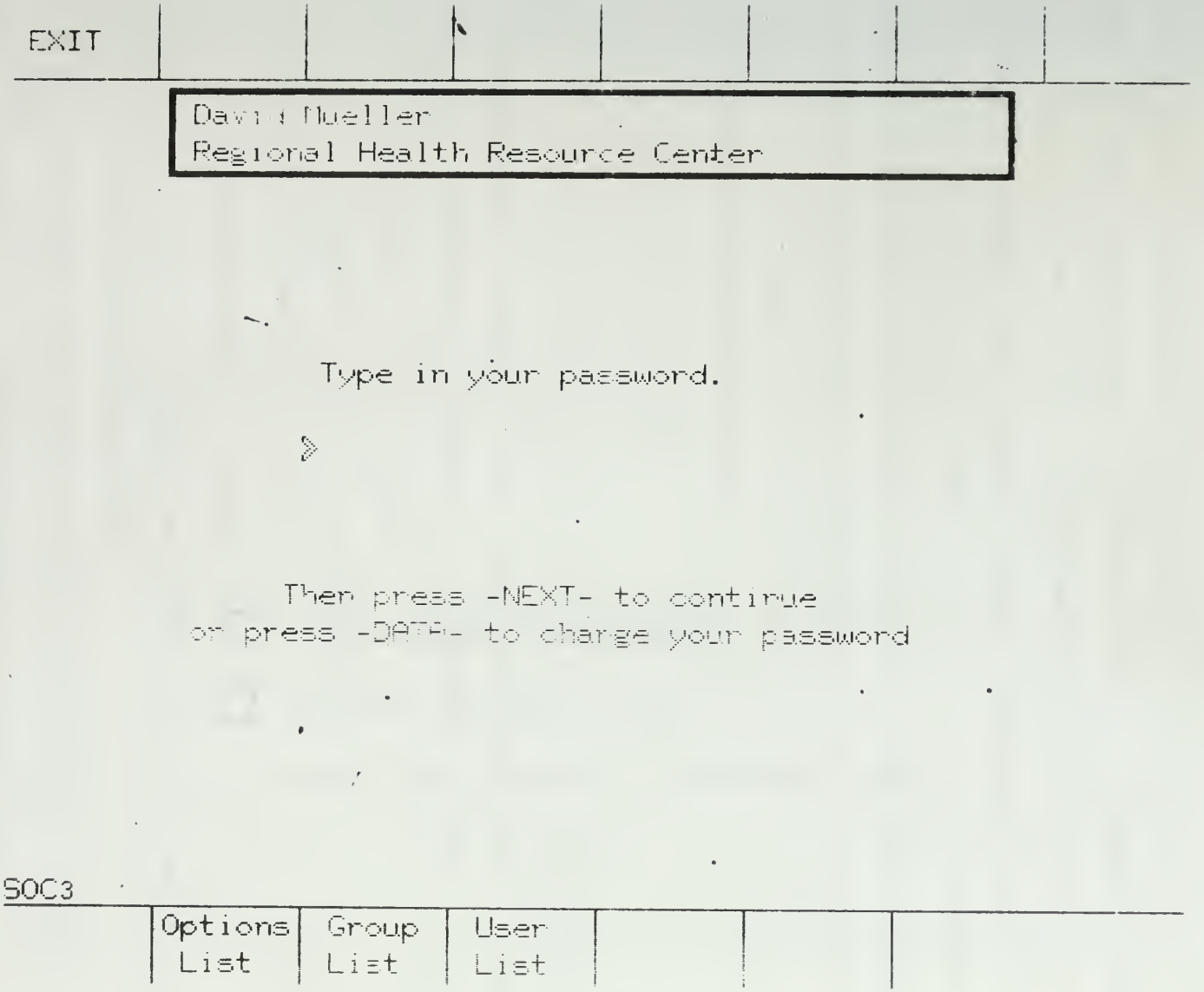

EXIT
David Mueller
Regional Health Resource Center
Type in your password.
Then press -NEXT- to continue
or press -DATA- to change your password
SOC3
Options List
Group List
User List

Figure 14

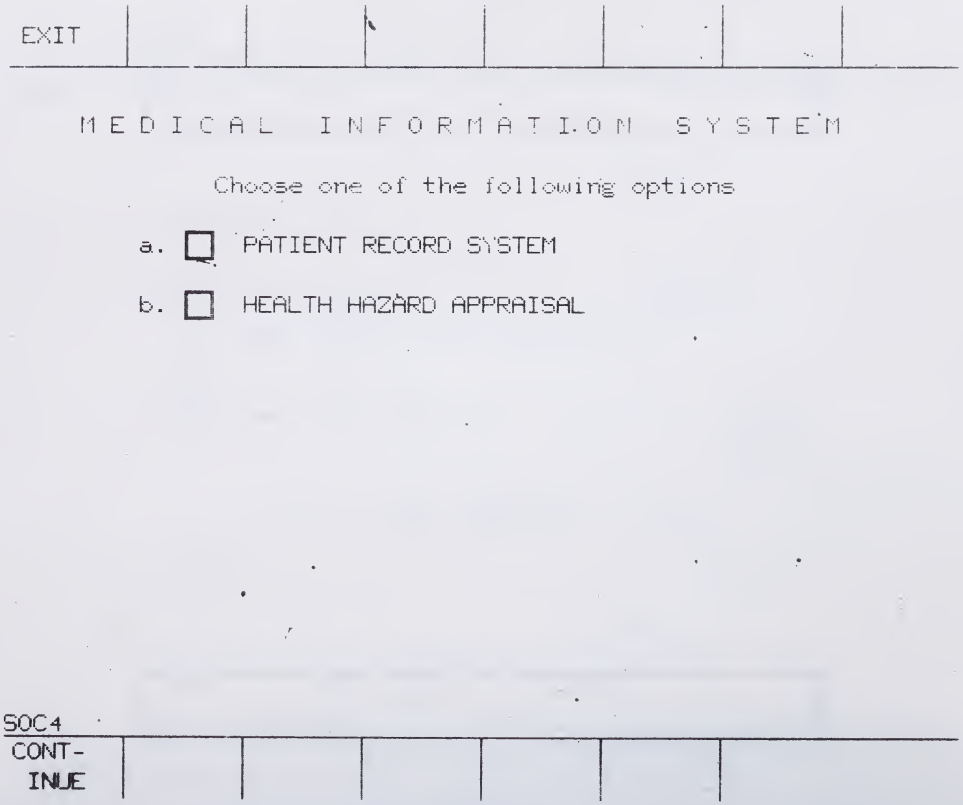

EXIT
MEDICAL INFORMATION SYSTEM
Choose one of the following options
a. PATIENT RECORD SYSTEM
b. HEALTH HAZARD APPRAISAL
SOC4
CONT-
INUE

Figure 15

and sensitive information. For security reasons this information belongs in the Master Directory entry for that patient. It is necessary to obtain this entry for any access to the patient record. In addition, there are certain functions available to the terminal user which involve only the Master Directory. The content of the Master Directory entry can be logically divided into three distinct sets of data: 1) the system control data, 2) the patient identifying data, and 3) the patient sensitive data. A discussion of the structure of the Master Directory will include a description of the scheme to address Master Directory entries and the organization of data within the entries.

7.1.1. System Control Information

Included in this subset of the entry are pieces of data which are needed by user routines for system chores, but which are not personal in nature. The following are fields which belong to this group:

Cipher key	60 bits
Access Control	3 bits per user
Link to item list	15 bits
Trace information	60 bits

The cipher key is used in ciphering the record by means of a method utilizing the logical difference operation.

The access control field is a three-bit pattern for each potential user group. The field assigns to each user group a type of access. Figure 16 lists the possible patterns and their meanings. Owner access allows the user to modify the patient record and, in addition, the access

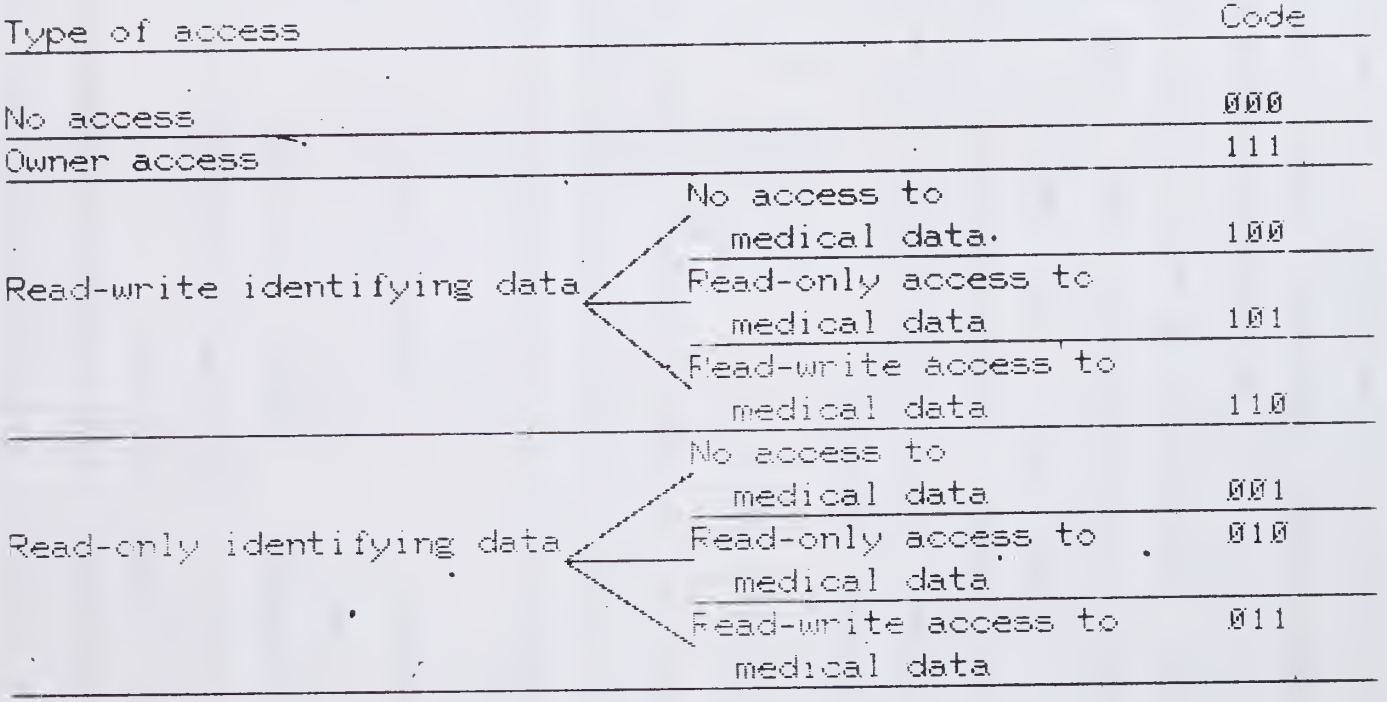

Type of access
Code
No access
000
Owner access
111
Read-write identifying data
No access to medical data.
100
Read-only access to medical data
101
Read-write access to medical data
110
Read-only identifying data
No access to medical data
001
Read-only access to medical data
010
Read-write access to medical data
011

Figure 16

control bits for other users of that record. Read-write access to any part of the record allows modification of that part by the user, the single exception being permanent data items.

The Item List is the backbone structure of the medical record for a particular patient and it is necessary that the Master Directory point ahead to this list connecting all parts of the patient record.

The trace information, for the purposes of this discussion, will be the user number of the creator of the record, the date and time of creation, the user number of the last user to access this record and the date and time of access. This field could become several words long if more information is needed.

Another link field in the system control information which contains a link to a special second page of sensitive data which may be defined at some future date. This data is medical data and is considered to be sensitive, but not of an identifying nature. This extension may also require some type of access control.

For a system with 20 groups, an assumption which will be maintained throughout this discussion, the system control information is four words long.

7.1.2. Patient Identifying Data

The second page of patient sensitive data will not be discussed further because its content is still largely undefined. Those items which tend to identify the patient, the so-called "identifying data," will be detailed here.

Several of the fields, such as the patient's name, contain variable-length information which is padded on the right with null characters. Figure 17 lists the fields belonging to the identifying data. These fields represent a maximum of 64 words. The exact arrangement of data is not a problem here since, as will be shown, there is some extra space, in fact, very little compaction is possible.

7.1.3. Master Directory Structure

A path to the patient record must begin in the Master Directory Index (Figure 18). This index is a list of the disk addresses of all Master Directory physical blocks. Since each physical block represents four entries, the index entry actually points to records 0, 4, 8, ... And addresses for other record entries must be computed as offsets into the physical block addressed by the preceeding multiple of 4. Specifically, the n^{th} element in the index list represents record number 4n and the address A of any record number i is found by,

$$A = \text{index}(\text{int}[i/4]) + 80\,(i - \text{int}[i/4]) \text{ words.}$$

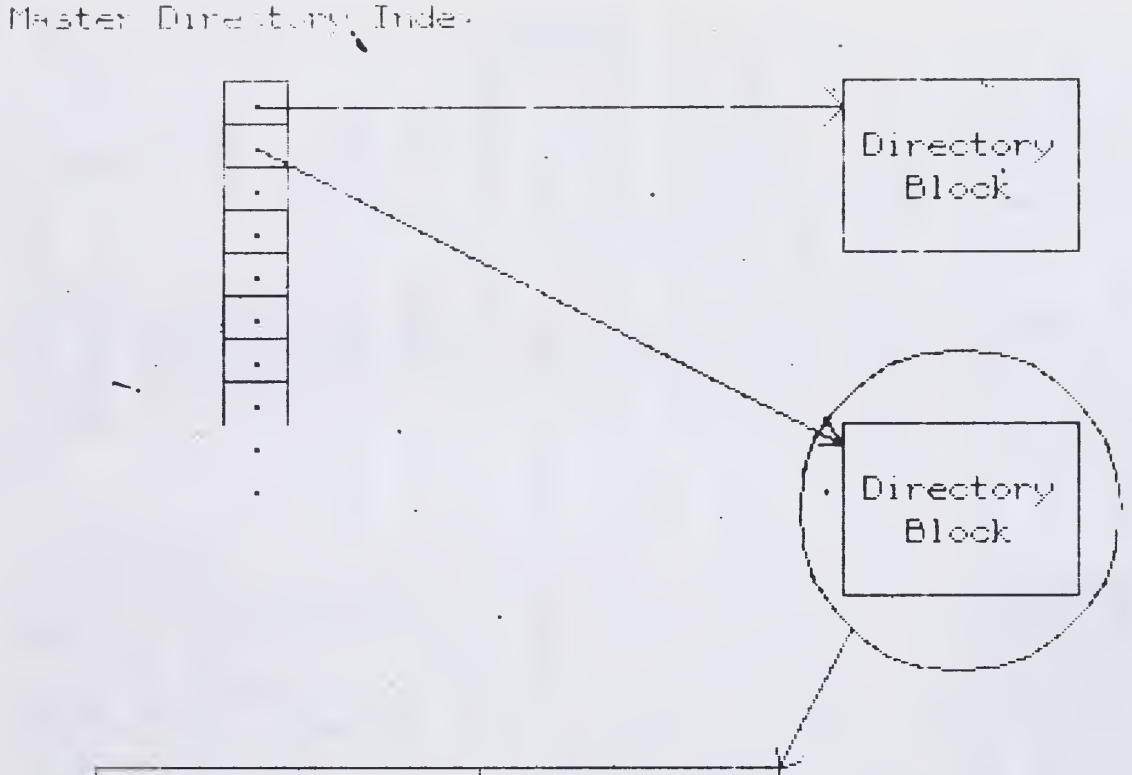
As was mentioned before, each 80 word entry is logically divided into a four-word system control block and 64 words of identifying data, leaving 12 words as yet unassigned. The Directory Index uses one word per four index entries and four records per index entry. Therefore, allowing for 1,000 patient records, the list will need less than 63 words.

7.2. The Master Directory Component Modules

The MDC may be viewed as a collection of modules each accessed

Data item	Length
Patient	
Name	3 words
Home address	6 words
Home Telephone	1 word
Business address	6 words
Business telephone	1 word
Birthdate	16 bits
Sex	2 bits
Occupation	1 word
Insurance company	3 words
Policy number	2.5 words
Social Security Number	30 bits
Number of children	4 bits
Comments	7 words
Next of kin	
Name	3 words
Home telephone	1 word
Business telephone	1 word
Relationship	1 word
For each of a maximum of 8 children	
Name	3 words
Birthdate	16 bits
Sex	2 bits

Figure 17



System Control Information	Identifying Data
System Control Information	Identifying Data
System Control Information	Identifying Data
System Control Information	Identifying Data

Figure 18

by interrupts. The interrupts are generated by touching the routing boxes at the top and bottom of the screen. Each module continues to perform its task until one of the interrupts is recognized. Depending on the module executing, the sequence in which it is executing, and the interrupt, control is passed to another module. The first module does not regain control until its interrupt is recognized. Each of the modules is discussed below.

7.2.1. PIN Request Page (see Figure 19)

This module is responsible for retrieving a Master Directory entry during the open patient record sequence. It obtains the PIN from the terminal user, reads the corresponding Master Directory entry, and checks the user's access rights. If an entry does not exist for this user or if the user does not have access, he is informed of the situation and allowed to either reenter the PIN or exit. If the entry does exist and the user does have access, then he is sent to the display data page. The user may also ask to create a patient record, in which case he is again routed to the brief ID page, except that the mode is changed to one of inputting data.

7.2.2. Brief ID Page Display (see Figure 20)

A subset of the patient identifying data is displayed to the user so that he may quickly confirm the record as the record he requested. The user may change any data element by touching the screen where the element appears and entering the change.

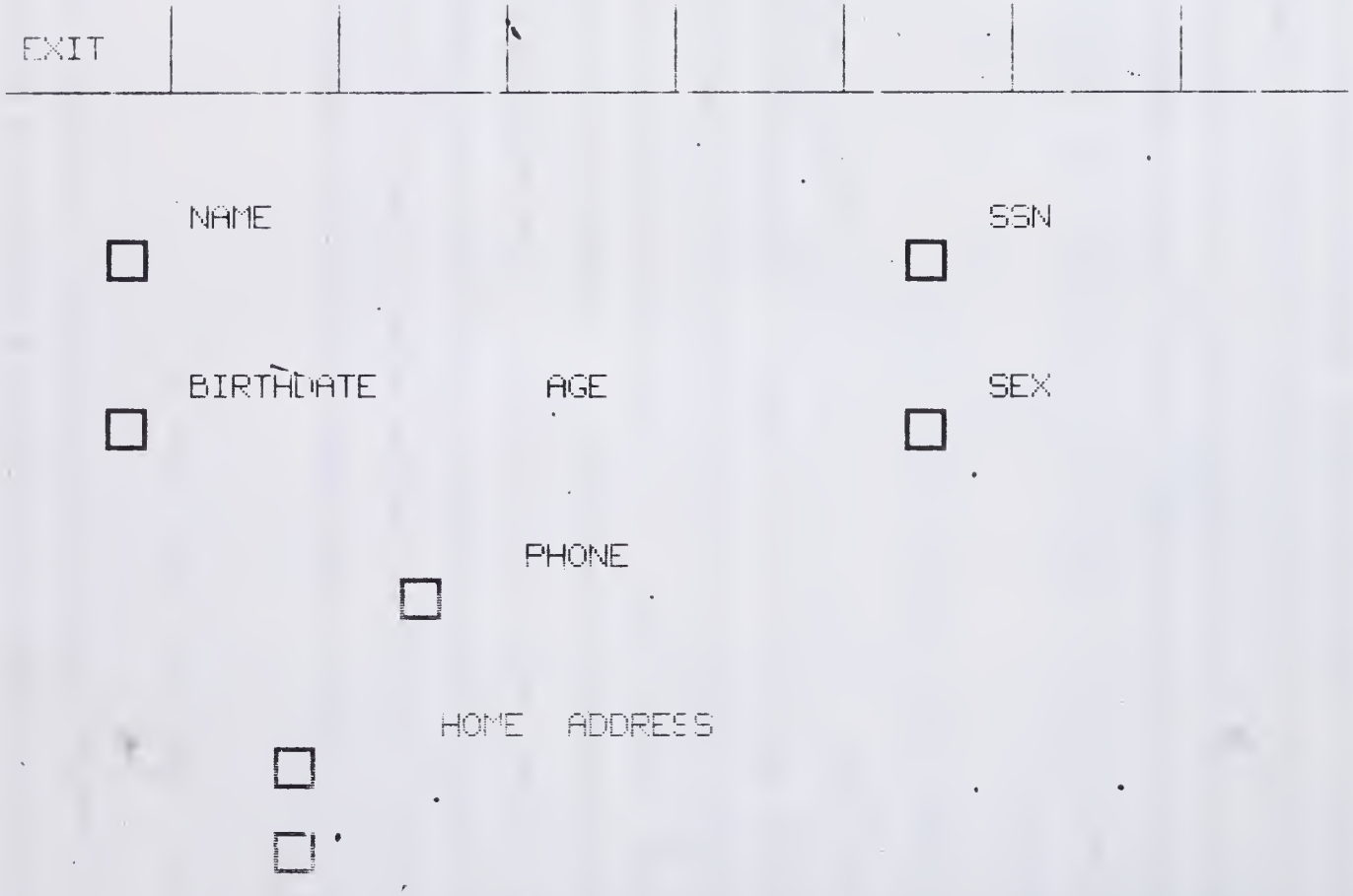

EXIT
NAME
SSN
BIRTHDATE
AGE
SEX
PHONE
HOME ADDRESS
Enter the patient index number >

Figure 19

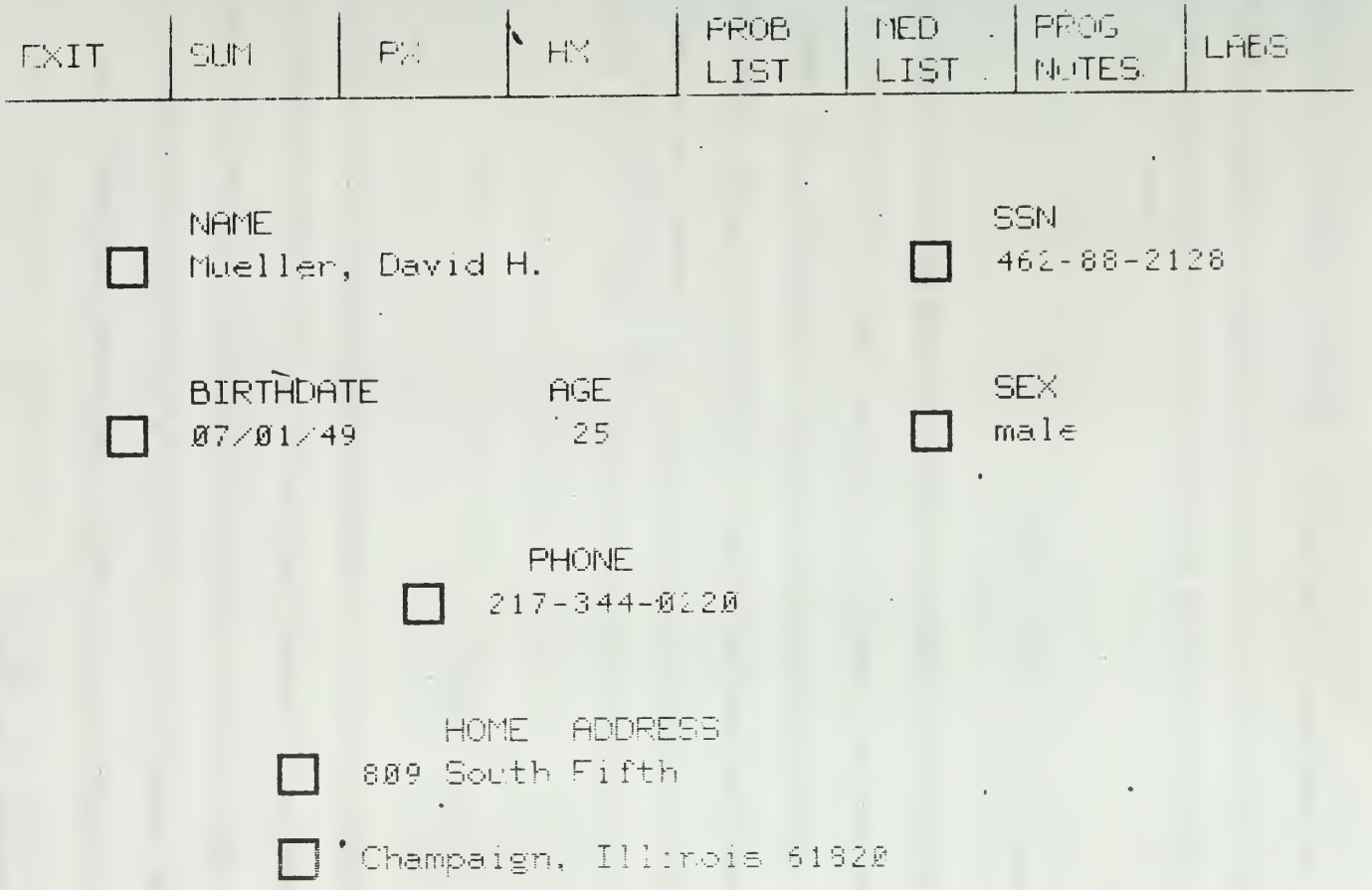

EXIT
SUM
PX
HX
PROB LIST
MED LIST
PROG NOTES
LABS
NAME
Mueller, David H.
SSN
462-88-2128
BIRTHDATE
07/01/49
AGE
25
SEX
male
PHONE
217-344-0120
HOME ADDRESS
809 South Fifth
Champaign, Illinois 61820

Figure 20

7.2.3. More ID Page Display

The remainder of the patient identifying data is displayed to the user for verification, input, or modification. Again, he may edit some data element by touching its position on the screen and then entering the change.

7.2.4. Change Access Rights Module

This module allows the user to change the access rights to the record if he has the right to do so. The user is given a list of the groups and their current access rights and allowed to alter these. He may also specify a new record owner. If he does so, the access template belonging to the new owner is copied into the record's access vector. The present user keeps his original level of access until he closes the record.

7.2.5. Store Data Module

When completing the creation of a patient record a PIN is assigned to the record and displayed to the user, and the Master Directory entry is written out. Space is obtained for the Item List and the Item List is initialized and written out. The user is then given the option of either opening the record or going back to the record assignment page.

7.2.6. Complete Open Module

When completing the open record sequence, the module first checks the access rights to the medical data. If access is granted, the routine

reserves control blocks. Finally, it sets the user's router variables and goes to the option selected by the user.

8. The Item Manager Component

8.1. Router Common

Router common is the basic communications area for the Record Manager. Here are kept all the data structures which allow applications lessons to access records once they have been opened. These data structures are divided into four general structures: 1) the Patient Directories (PD), 2) the User Banks (UB), 3) the Item Data Buffers (IDB), and 4) the System Status and Statistics Tables (see Figure 21).

8.1.1. The Patient Directory

One PD exists for each record which is open (see Figure 22). It is initialized by the MDC, and contains record status information and the Record's Block List and Item List. All of this is read from the disk. The status information includes the record's cipher key (currently not used), pointers to the lists of free space in the record, the length of the free space lists, and the actual number of items present (see Figure 23). The Block List serves as a translation table between the logical block numbers stored with the other item information and the physical block numbers required to read the data. It contains one entry for each physical block assigned to the record; each 15-bit entry gives the physical block number associated with the corresponding logical block.

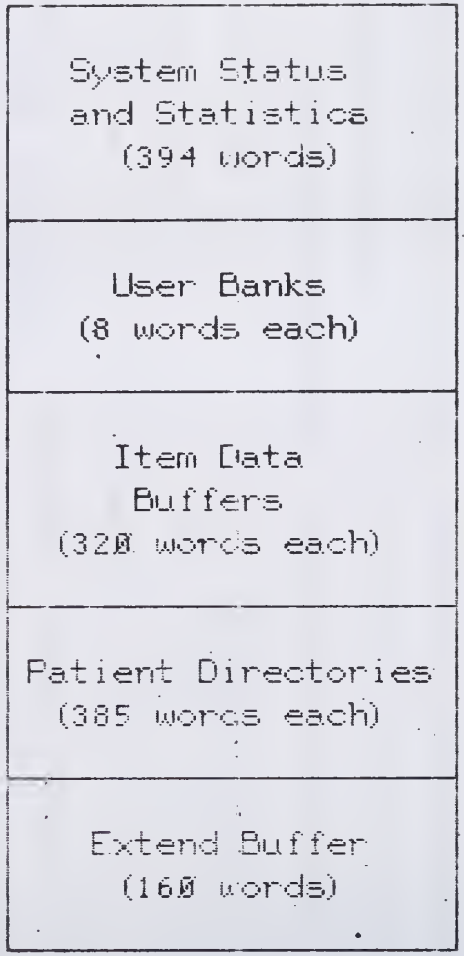

System Status
and Statistics
(394 words)
User Banks
(8 words each)
Item Data
Buffers
(320 words each)
Patient Directories
(385 words each)
Extend Buffer
(160 words)

Figure 21

Record Status
(4 words)
Block List
(20 Words)
Item List
(296 words)
Attach Flags
(60 words)
Attach Bits
(5 words)

Figure 22

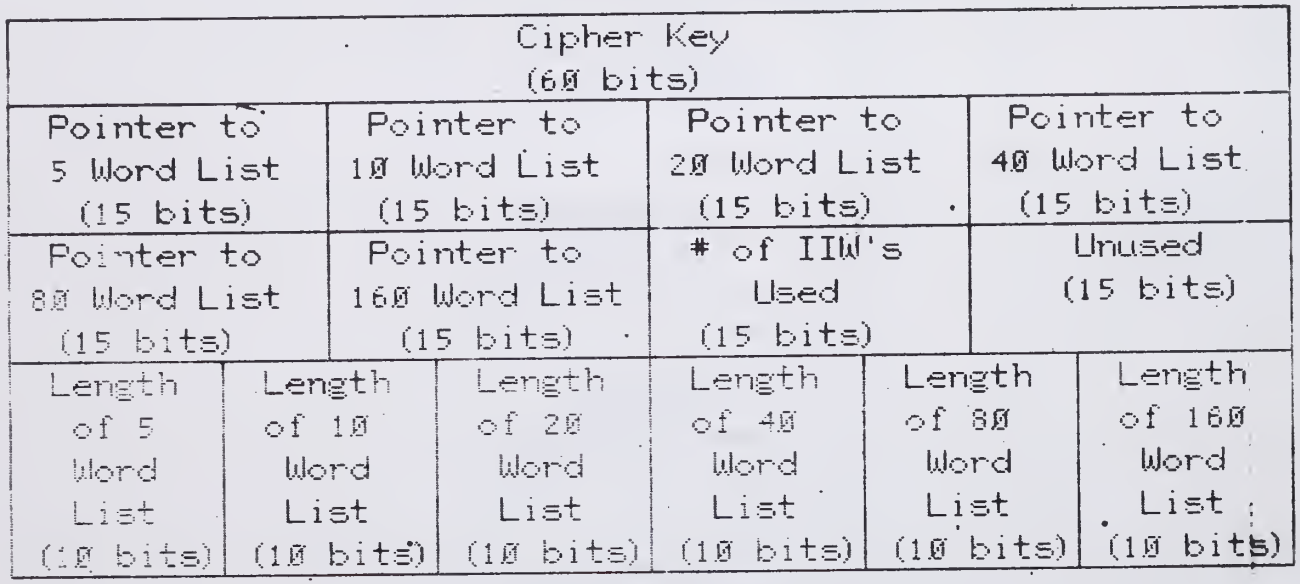

Cipher Key
(60 bits)
Pointer to 5 Word List (15 bits)
Pointer to 10 Word List (15 bits)
Pointer to 20 Word List (15 bits)
Pointer to 40 Word List (15 bits)
Pointer to 80 Word List (15 bits)
Pointer to 160 Word List (15 bits)
# of IIW's Used (15 bits)
Unused (15 bits)
Length of 5 Word List (10 bits)
Length of 10 Word List (10 bits)
Length of 20 Word List (10 bits)
Length of 40 Word List (10 bits)
Length of 80 Word List (10 bits)
Length of 160 Word List (10 bits)

Figure 23

The Item List contains one Item Information Word (IIW) for each item (see Figure 24).

In addition to this permanent data on the record, there are two tables for information valid only when the record is open. These tables contain information to prevent more than one user from modifying the same item at the same time. The function which accomplishes this is known as an attach. An item which is attached by one user may not be written by another. In addition, to guarantee that the user attaching the item has the latest copy of it, he is required to read the item after attaching it but before writing it.

The first table contains in each 12-bit entry, the station number + 1 to which the item is attached, 0 meaning that the tiem is not attached. The second table contains 1 bit per entry. This bit is set to 1 when the item is read after it is attached. The bit is then checked before a write is attempted, and the write is not allowed if the bit is not set.

8.1.2. The User Bank

One UB is assigned for each station signing into the SOC (see Figure 25). The UB is used both for status information on the user and communications between the IMC and USC. The first word contains status information. The second contains the user's request if a request for service from the IMC is being made, or, if the request has been satisfied, the return result code (see Figure 26). The remaining words are used for the request parameters.

Item Name (30 bits)	Item Type (4 bits)	Logical Block Number (7 bits)	Position in Block ÷5 (6 bits)	Number of Item Accesses (13 bits)

Figure 24

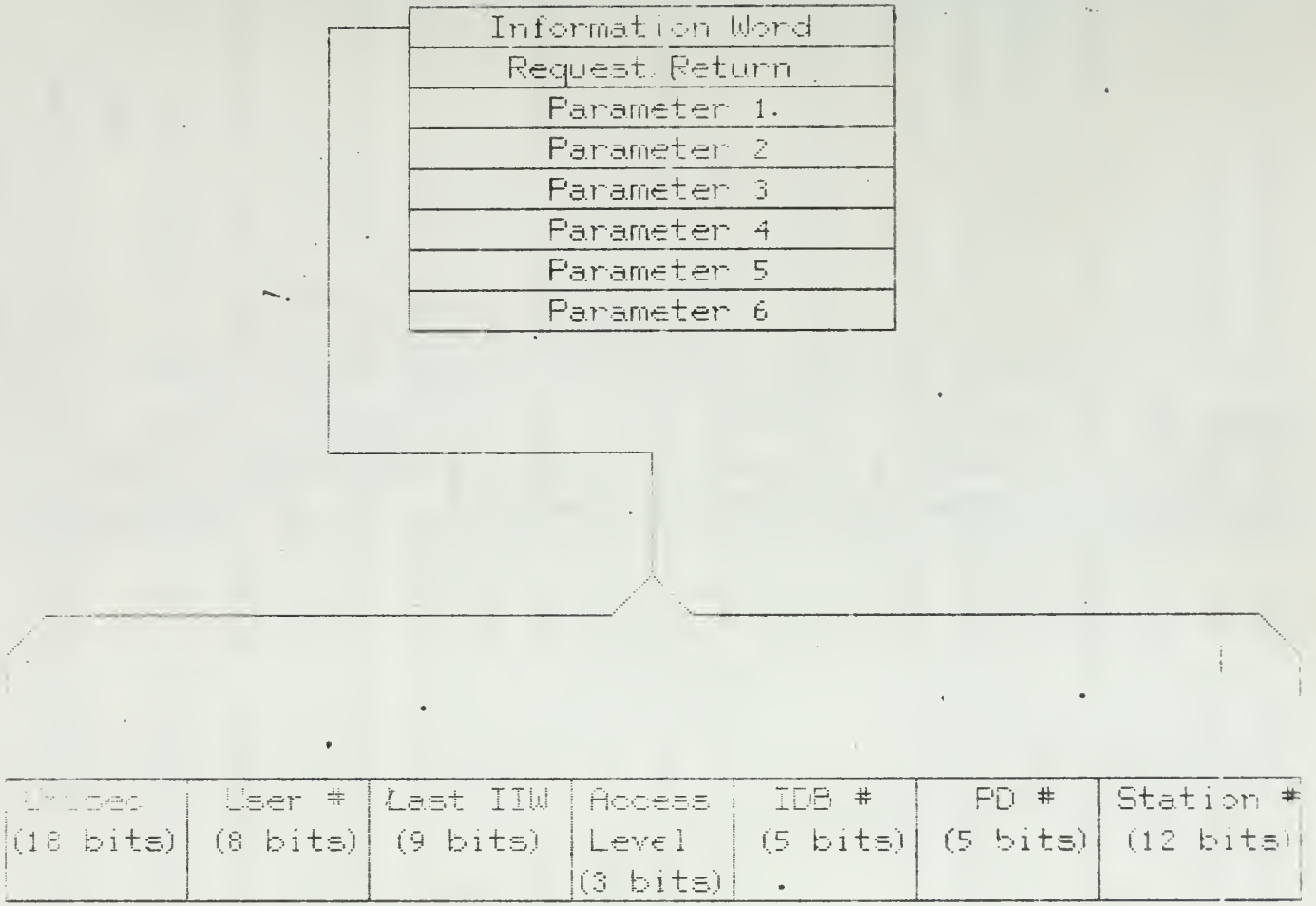

Information Word
Request Return
Parameter 1.
Parameter 2
Parameter 3
Parameter 4
Parameter 5
Parameter 6
Unused (18 bits)
User # (8 bits)
Last IIW (9 bits)
Access Level (3 bits)
IDB # (5 bits)
PD # (5 bits)
Station # (12 bits)

Figure 25

Function Code (12 bits)	# of Arguments (4 bits)	Unused (8 bits)	Length of Lesson Common (13 bits)	Length of Lesson Storage (11 bits)	Station # +1 (12 bits)

Figure 26

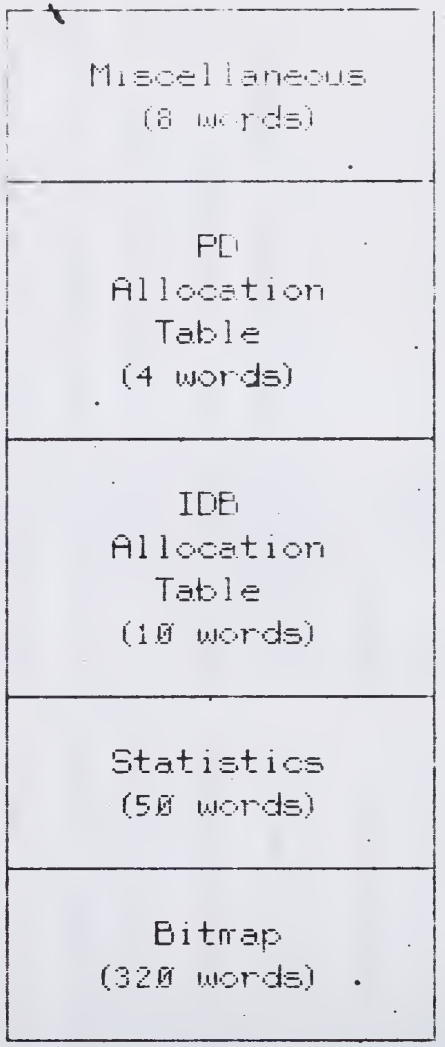
8.1.3. The Item Data Buffers

The IDBs are used as I/O buffers. When a block is needed from disk, the IDB allocation table is searched. If the block is found, no read is necessary. Otherwise, the allocation table is searched to find an empty IDB, and, if none is found, the one currently used by the station making the request is used. In either case, the block is read into storage and put in the IDB selected. Since one IDB may be assigned to each user, there must be at least as many IDBs as there are UBs.

8.1.4. The System Status Information

The System Status Information includes IDB and PD allocation tables, the disk allocation bitmap, a buffer used to contain items while they are being extended, and miscellaneous information (see Figure 27). The miscellaneous information includes the station number of the pseudo-terminal and looks for the initialization routines. These are used to prevent users from entering the system before initialization is complete. The IDB and PD allocation tables are both 15-bit segmented vectors. Entries in both tables are pointers to the disk block from which the corresponding buffer was filled. The bitmap contains one bit for each block available in the database. The bit is 0 if the block has not been assigned and 1 if it has.

Statistics are kept to measure system performance. Current data collected are listed in Figure 28.



Note - extend buffer follows PD's

Figure 27

1. Number of reads
2. Number of writes
3. Number of bitmap returns
4. Number of PD returns to disk
5. Number of PD reads from disk
6. Number of users
7. CPU time used
8. Elapsed run time
9. Number of new PD buffer requests
10. Number of record closes
11. Number of times each IMC request used

Figure 28

8.1.5. Length of Router Common

The length of router common may be calculated from the following formulas:

Common length = Length of system status and statistics + Number of UBs x (Length of UB + Length of PD + Length of IDB) + (Number of IDBs - Number of UBs) x Length of IDB.

Length of system status and statistics = Overhead + Length of PD allocation buffer + Length of IDB allocation table + Length of statistics + Length of bitmap + Length of extend buffer.

Length of UB = UB overhead + Number of parameters.

Length of PD = PD overhead + Length of Block List + Length of Item List + Length of Attach Tables.

Length of Block List = Floor (Number of blocks in list x Block List Entry Length) + 1.

Length of Item List = Number of items x Length of IIW.

Length of Attach Tables = Ceil (Number of items x (Length of Attach Flag + Length of Attach Bit)).

Current values of these constants are given in Figure 29. Note that the following relation must hold:

# of IDBs $\geq$ # of UBs.

8.2. Router Variables

Several parameters are stored in router variables. These variables are available to the applications lesson and to the USC, and are listed in Figure 30.

```
Attach Flag Length              = 1/5  word
Length of Attach Bit            = 1/60 word
# of Items                      = 296
Length of Attach Table          = 65   words
Length of IIW                   = 1    word
Length of Item List             = 296  words
Length of Block List Entry      = 1/4  word
# of Blocks in Block List       = 78
Length of Block List            = 20   words
Length of PD Overhead           = 4    words
Length of PD                    = 385  words
Length of UB overhead           = 2    words
# of Parameters                 = 6    words
Length of UB                    = 8    words
Length of Status Overhead       = 8    words
Length of Extend Buffer         = 160  words
Length of PD Allocation Table   = 4    words
Length of IDB Allocation Table  = 10   words
Length of Statistics            = 50   words
Length of Bitmap                = 322  words
Length of System Status         = 554  words
Length of IDB                   = 320  words
# of UB's                       = 3
# of IDB's                      = 3
Length of Common                = 2693 words
```

Figure 29

Words	Description	Length
1	Group Number	8 Bits
2	User Number	8 Bits
3	Absolute Location of User Bank	60 Bits
4-7	Patient ID String	40 Characters
8	Patient Sex 0=Unknown 1=Male 2=Female	2 Bits
9-11	Patient Name	30 Characters
12	Patient Birthdate	8 Characters
13-14	Patient Social Security Number	11 Characters
15	Patient Index Number	10 Bits
16	Access Level to Record -1=Read-write 0=Read Only	1 Bit

Figure 30

8.3. The Item Manager Modules

The IMC consists of three sets of modules. The first set handles the interrupts, the second is a set of service routines for the component, and the third is the set of routines which accomplishes the functions of the component. Each of these sets is explained in greater detail below.

8.3.1. The Interrupt Service Set

This set of modules is the main driver for the component. It consists of two modules, Interrupt and Service. The former recognizes the presence of an interrupt and the latter determines its nature and calls the appropriate module to service the interrupt. Upon completion of the service, control returns to the Interrupt Module.

8.3.2. The General Service Set

This set consists of modules which provide services to other modules in the component. Included are the Space Allocation and Deallocation Modules, the Item List Search Module, and the I/O Buffer-Router Common Transfer Module. These are explained in greater detail below. The component also uses the Bitmap Allocation and Deallocation and Database Read and Write routines.

The Free Space Allocation Module is responsible for allocating space of a given size for items. This is done with a modification of the "buddy" system. The module searches for a block of free space of

the requested size. If such a block is not present, it finds the smallest block available which is large enough to satisfy the request and splits it until the appropriate size is reached. The Free Space Deallocation Module simply places the block on the appropriate free space list. Both of these modules may call the Allocate Module if a 320-word block is desired.

The Item List Search Module locates the given IIW in the Item List. It uses the parameters passed in the interrupt to determine the direction of the search, whether forward or backward, and the starting point, whether beginning end, or from point last searched. The actual name found is placed in the system variable "iname."

The I/O Buffer-Router Common Transfer Module is responsible for transferring data from the router common data buffer to the storage I/O buffer and vice-versa.

The Allocate Routine searches the bitmap to find a free block. The Database Read and Write routines read and write disk blocks, transferring their contents to and from router common.

8.3.3. The Functional Set

This set consists of modules which actually perform the functions of the IMC. They call modules from the service set as needed to accomplish these tasks. There is one module for each of the following functions: Get Item, Put Item, Create Item, Delete Item, Extend Item, Attach Item, Detach Item, and Replace Item. They all use parameters provided in the UB, and set their return parameters there.

The Get Item Module reads in a given item. It calls the Item List Search module to locate the IIW and the Database Read Module to read the item into the I/O buffer. The Put Item Module reverses the process, except that a check is made first to determine whether or not the item is attached to this user and whether or not he has read it since the attach. The Replace Item Module performs operations equivalent to the sequence: Attach Item, Get Item, Put Item, Detach Item. The difference is that the Detach Item is done only if the item was not attached before the Replace Item was begun.

The Create Item Module calls the Allocate routine to obtain free space for the new item, creates a new IIW, calls the Database Read to read the item, if necessary, and the Database Write to write the item out. The Delete Item Module checks to see that the item is temporary, calls the Item List Search Module to find the IIW, removes the IIW from the Item List, and calls the Free Space Deallocation Module to mark the space as available in the bitmap.

The Extend Item Module calls the Item List Search routine to locate the IIW, checks to see that the item is rewritable and attached to this user, calls the Allocate Routine to get space for the new item size, modifies the IIW to point to the new spaces, calls the Database Read routine to read the new space, transfers the item to the new space, calls the Database Write routine to write the new space out, and releases the old space.

The Attach Item Module first looks at the user's access level to determine if he has read-write access to the medical data. If so,

the module calls the Item List Search Module to locate the IIW and marks the item as attached if it is not already attached. The Detach Module calls the Item List Search Module to locate the IIW and marks the item as detached if it was attached to this user.

For all functions except Attach Item and Detach Item, the following information is set up for transfer to MIS system reserved words:

1. Result code for the function
2. Item name last referenced
3. User ID of the creator of the item
4. Date of creation
5. Time of creation to the half day
6. User ID of the last user to write the item
7. Date of the last write
8. Time of the last write to the minute

In the case of the Attach Item and Detach Item functions, only the first two of these are set; the remaining six are zeroed.

9. The User Service Component

The User Service Component consists of two sets of modules. The first is the interface to the IMC and the second is a set of service modules which use data produced by the Record Manager. These modules are library routines. The source code is in the MIS system library, and the condensed code is in the applications lesson.

9.1. Record Manager Interface Modules

This set contains the modules called by the applications program to request Item Manager Service. There is a separate module for each function. The correspondence between functions and modules is given in Figure 30.

The interface set also contains modules called by the interface modules. The interrupt module transfers the parameters to and from router common and sends and receives the interrupts. Two transfer modules transfer data in and out of the router common buffers. The item length module locates the item length for the put item module, and the disk error routine notifies the user of disk errors occurring in the IMC.

9.2. User Service Modules

The modules in this set provide miscellaneous service to the applications programs. The ID box module displays the patient ID in the lower right hand corner of the screen. The last name, initial, birthdate and sex are displayed in positions 3153 to 3164 and 3253 to 3264.

BLIOGRAPHIC DATA IEET	1. Report No. UIUCDCS-R-75-751	2.	3. Recipient's Accession No.
Title and Subtitle MEDICAL DATA MANAGEMENT			5. Report Date August, 1975 6.
Author(s) Robert A. Rush			8. Performing Organization Rept. No.
Performing Organization Name and Address University of Illinois at Urbana-Champaign Department of Computer Science Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
. Sponsoring Organization Name and Address University of Illinois at Urbana-Champaign Department of Computer Science Urbana, IL 61801			13. Type of Report & Period Covered M.S. Thesis - 1975 14.

. Supplementary Notes

. Abstracts

Record keeping in a medical environment is discussed and a proposal is made for the organization of data in a medical database. A data description language is presented to facilitate the organization, and processes for data manipulation are described. Specifications for the data management system in use by the MIS are included.

. Key Words and Document Analysis. 17a. Descriptors

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page UNCLASSIFIED	22. Price

IRM NTIS-35 (10-70)

USCOMM-DC 40329-P71

OCT 10

OCT 2 3 1973

BOUND BY HNM

UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 no.746-751(1974
Lisp a CAI implamentation /
3 0112 088402075